Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations 1. Thesis and Dissertation Collection, all items

1998-03-01

Software system requirements for the fuel automated subsystem of the Integrated Combat Service Support System (ICS3) using the Computer Aided Prototyping System (CAPS)

Kominiak, Lawrence A.

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/26777

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

Downloaded from NPS Archive: Calhoun

DUDLEY KNOX LIBRARY

http://www.nps.edu/library

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

NPS ARCHIVE
1998.03
KOMINIAK, L.

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

SOFTWARE SYSTEM REQUIREMENTS FOR THE FUEL AUTOMATED SUBSYSTEM OF THE INTEGRATED COMBAT SERVICE SUPPORT SYSTEM (ICS3) USING THE COMPUTER AIDED PROTOTYPING SYSTEM (CAPS)

by

Lawrence A. Kominiak

March 1998

Thesis Advisor: Luqi

Approved for public release; distribution is unlimited.

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1998	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE: SOFTWARE SYSTEM REQUIREMENTS FOR THE FUEL AUTOMATED SUBSYSTEM OF THE INTEGRATED COMBAT SERVICE SUPPORT SYSTEM (ICS3) USING THE COMPUTER AIDED PROTOTYPING SYSTEM (CAPS)	5. FUNDING NUMBERS
6. AUTHOR(S) Kominiak, Lawrence A.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT: Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE:

13. ABSTRACT *(maximum 200 words)*

The United States Army is currently developing and testing Force XXI, an attempt to redesign itself by the early years of the 21st century to incorporate digital technology and advanced weaponry. In 1996, the United States Training and Doctrine Command mandated that all combat service support disciplines be automated to the greatest extent possible. Concurrently, the Deputy Chief of Staff for logistics, United States Materiel Command, and the Combined Arms Support Command (CASCOM) developed a future strategic vision of seamless logistics support. To support this vision, CASCOM has proposed the implementation of the Integrated Combat Service Support System (ICS3) as the Army's single seamless combat service support management system. ICS3 will be a "system of systems" that automates the combat service support disciplines of man, arm, fuel, fix, move, and sustain. Specifically, the combat service support discipline of fuel will be incorporated in ICS3 as the Fuel Automated Subsystem.

This thesis analyzes current Army petroleum operations, identifies petroleum accountability/management procedures as the target domain for automation, and develops the respective software system requirements. From the software system requirements, a prototype for the Fuel Automated Subsystem is successfully developed using the Computer Aided Prototyping System (CAPS) to illustrate the system's viability.

14. SUBJECT TERMS CAPS, System Analysis, Software Requirements, Prototyping, ICS3, Fuel Automated Subsystem	15. NUMBER OF PAGES 302
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited

SOFTWARE SYSTEM REQUIREMENTS FOR THE FUEL AUTOMATED SUBSYSTEM OF THE INTEGRATED COMBAT SERVICE SUPPORT SYSTEM (ICS3) USING THE COMPUTER AIDED PROTOTYPING SYSTEM (CAPS)

Lawrence A. Kominiak
Major, United States Army
B.S., United States Military Academy, 1987

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

March 1998

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

ABSTRACT

The United States Army is currently developing and testing Force XXI, an attempt to redesign itself by the early years of the 21st century to incorporate digital technology and advance weaponry. In 1996, the United States Training and Doctrine Command mandated that all combat service support disciplines be automated to the greatest extent possible. Concurrently, the Deputy Chief of Staff for Logistics, United States Materiel Command, and the Combined Arms Support Command (CASCOM) developed a future strategic vision of seamless logistics support. To support this vision, CASCOM has proposed the implementation of the Integrated Combat Service Support System (ICS3) as the Army's single seamless combat service support management system. ICS3 will be a "system of systems" that automates the combat service support disciplines of man, arm, fuel, fix, move, and sustain. Specifically, the combat service support discipline of fuel will be incorporated in ICS3 as the Fuel Automated Subsystem.

This thesis analyzes current Army petroleum operations, identifies petroleum accountability/management procedures as the target domain for automation, and develops the respective software system requirements. From the software system requirements, a prototype for the Fuel Automated Subsystem is successfully developed using the Computer Aided Prototyping System (CAPS) to illustrate the system's viability.

TABLE OF CONTENTS

I. INTRODUCTION 1
A. GENERAL BACKGROUND 1
B. PURPOSE 2
C. RAPID PROTOTYPING & CAPS 2
D. METHODOLOGY & DELIVERABLES 3
E. ORGANIZATION OF THESIS 4
II. BACKGROUND OF THE FUEL AUTOMATED SUBSYSTEM 5
A. INTRODUCTION. 5
B. FORCE XXI 5
1. Changing Missions and the Future Force 5
2. Evolution of Doctrine and Force Structure 7
3. Characteristics of Force XXI Operations 8
4. Leveraging Information Technology 9
C. FORCE XXI COMBAT SERVICE SUPPORT 10
1. Combat Service Support Defined 10
2. Logistical Objectives 11
D. INTEGRATED COMBAT SERVICE SUPPORT SYSTEM 12
1. Current CSS Structure (Pre-Integrated Combat Service Support System) 12
a. Procedures and Operating Methods 12
b. Information Flow 13
c. Security 15
2. Integrated Combat Service Support System Overview 16
3. Development of the Integrated Combat Service Support System 16
4. Integrated Combat Service Support System End State Vision 20
E. CURRENT ARMY FUEL SYSTEM PROCEDURES & AUTOMATION 22
1. Petroleum Product Classifications and Retail Operations 23
2. Manual Fuel Management 23
3. Automated Fuel Management 26
III. COMPUTER AIDED PROTOTYPING SYSTEM 31
A. INTRODUCTION 31
B. THE PROTOTYPING PROCESS 31
1. Traditional Software Development 31
2. Rapid Prototyping 33
C. CAPS DEVELOPMENTAL ENVIRONMENT 35
1. Prototype System Description Language 35
a. Operators 36
b. Streams 36
c. Types 37
d. Timing Constraints 37
e. Control Constraints 37
2. Additional CAPS Resources 38

IV. REQUIREMENTS ANALYSIS AND MODELING 39
A. INTRODUCTION 39
B. REQUIREMENTS ANALYSIS 39
1. System Environment 43
2. System Goals 44
3. System Constraints 46
4. Model Summary 47
C. PROTOTYPE DESIGN 49
1. Modules 49
2. Basic Model 52
D. CAPS PROTOTYPE 54
1. Graphical Editor 54
2. Prototype Description Language 59
3. Translating and Scheduling 59
4. Implementation 60
V. CONCLUSIONS AND RECOMMENDATIONS 63
A. SIGNIFICANCE OF THE PROTOTYPE 63
B. RECOMMENDED FUTURE WORK 64
APPENDIX A MANUAL FUEL MANAGEMENT DOCUMENTS 67
APPENDIX B ENVIRONMENT MODEL 71
APPENDIX C HEIRARCHY OF SYSTEM GOALS 77
APPENDIX D CAPS GRAPHS 81
APPENDIX E PSDL SOURCE CODE 89
APPENDIX F ADA SOURCE CODE 113
APPENDIX G GRAPHICAL USER INTERFACES 279
LIST OF REFERENCES 285
BIBLIOGRAPHY 287
INITIAL DISTRIBUTION LIST 289

I. INTRODUCTION

A. GENERAL BACKGROUND

The United States Army is currently developing and testing Force XXI, an attempt to redesign itself by the early years of the 21st century to incorporate digital technology and advanced weaponry. Accordingly, over the next five years the Army is scheduled to devote approximately one-third of its research, development and acquisition budget ($28 billion of $87 billion) to solely support these digital technology initiatives [Ref. 1]. The main proponent for Force XXI is the United States Army Training and Doctrine Command (TRADOC), headquartered at Fort Monroe, Virginia. TRADOC is directly responsible for designing the tactical units of the future and developing/writing the doctrine by which the Army will fight. In February 1996, TRADOC published Pamphlet 525-5, Force XXI Operations, which mandated that the combat service support disciplines of man, arm, fuel, fix, move and sustain be automated to the greatest extent possible. Concurrently, the strategic logistics vision underpinning the combat service support system of the future was jointly developed by the Deputy Chief of Staff for Logistics (DCSLOG), the United States Army Materiel Command (AMC), and the Combined Arms Support Command (CASCOM). Their future strategic logistics vision is one of "seamless support: a continuum of support consisting of soldiers and civilians, business practices, and an intelligent information network of interrelated systems that provides world class support to the Army across the entire spectrum of military operations" [Ref. 2]. The concept of seamless support encompasses an integrated effort to streamline and unify the current combat service support system that is complex. The

objective of seamless support is to allow the war-fighter to focus on planning and executing the battle while freeing him from the complexities of combat service support (CSS) operations and systems.

To support the future combat service support vision, CASCOM has proposed the implementation of the Integrated Combat Service Support System (ICS3) as the Army's single, seamless, integrated and interactive CSS automated management system. ICS3 will be a "system of systems" incorporating all of the CSS disciplines of manning, arming, fueling, fixing, moving and sustaining the force. Specifically, the CSS discipline of fueling will be incorporated in ICS3 as the Fuel Automated Subsystem. Ultimately, CASCOM plans to have a contractor develop the Fuel Automated Subsystem as part of the ICS3 developmental process.

B. PURPOSE

The objective of this thesis is to analyze current petroleum operations, to identify a target domain for automation as the Fuel Automated Subsystem, and to develop the respective software system requirements. From the software system requirements, a prototype of the Fuel Automated Subsystem is modeled using the Computer Aided Prototyping System (CAPS) to illustrate its viability prior to establishing a contract for full subsystem development.

C. RAPID PROTOTYPING & CAPS

The use of prototyping in hardware engineering has been widely accepted, but remains relatively underutilized in software development. The traditional software development methodology dictates that a new system be tested near the completion of a project. As a result of the late testing in the traditional approach, errors are found late in

the development cycle. This untimely discovery of errors typically leads to a poor quality product (unreliable and non-maintainable software), late product delivery, and cost overrun.

Conversely, prototyping is an approach to software development and evolution that insures that the system's design meets proposed requirements while it is still in the analysis and design stages of development. A prototype is simply an executable test version of the proposed software system. The prototype is explicitly used to gain further information that can guide analysis and design, and can support automatic generation of production code. Prototyping is very appropriate for systems that lack a specific requirements definition. Systems without a strong requirements definition characteristically require several iterations of the prototyping process, each verified by the user for correctness/usability, prior to reaching the final production code.

Since the Fuel Automated Subsystem is a new and novel system that is not specifically defined, it lends itself well to the prototyping methodology. Computer assistance is required to rapidly construct a prototype for the Fuel Automation Subsystem. CAPS provides a set of integrated software tools which use a fifth generation language for automated software prototype development.

D. METHODOLOGY & DELIVERABLES

The top down approach is the general methodology used to analyze petroleum operations, identify a target domain for automation, requirement analysis, and prototype development. Specifically, the approach consists of initially analyzing Army Regulations, Department of the Army Pamphlets, and Field Manuals that describe the current Army fuel system and its procedures to identify a target domain for automation as

the Fuel Automated Subsystem. From this initial analysis, the Army's fuel system is decomposed into three distinct functional categories: operational functions, personnel functions and equipment functions. Since personnel and equipment functionality are incorporated in other modules of ICS3, the operational category is further decomposed to ultimately identify petroleum accountability as the target domain for automation. With the target domain identified, a requirement analysis establishes essential system functionality, attributes, and constraints. Based upon the requirements, CAPS is used to develop a prototype of the Fuel Automated Subsystem.

The deliverables are a requirements analysis for the Fuel Automated Subsystem, the corresponding prototype, and the thesis. The prototype is available to CASCOM to illustrate the viability of the Fuel Automated Subsystem.

E. ORGANIZATION OF THESIS

This thesis is divided into four chapters. Chapter I introduces the Fuel Automated Subsystem in the context of the Army's efforts to develop ICS3 as part of Force XXI, outlines the objective of the thesis, describes the role of CAPS, and addresses the general methodology used in the prototype development. Chapter II provides an overview of the development of Force XXI, combat service support in Force XXI, the role of ICS3, and the current Army fuel system's structure. Chapter III details the Computer Aided Prototyping System. Chapter IV describes the analysis of Army petroleum operations, identifies petroleum accountability as the target domain, and develops the requirement analysis process and architectural design of the Fuel Automated Subsystem prototype. Finally, Chapter V provides conclusions and recommendations for future follow-on work.

II. BACKGROUND OF THE FUEL AUTOMATED SUBSYSTEM

A. INTRODUCTION

In order to fully understand the role and requirements of the Fuel Automated Subsystem within the Integrated Combat Service Support System (ICS3), one must also understand the major structural changes that the Army is undergoing as it redesigns itself for the 21st century. The first section of this chapter introduces the Army's future force - Force XXI. The discussion includes a description of the changing mission environment and doctrine that has led to the development of Force XXI. Subsequent discussion focuses on the characteristics of future Force XXI operations and includes the role of information technology. The second section describes the objectives of combat service support operations, which adequately support Force XXI. One of the objectives discussed is the requirement for a single, seamless logistics system that leverages digital technology. The third section specifically describes the current combat service support structure, its shortfalls, and the development plans for ICS3 to meet the single, seamless logistics system requirement. The final section describes the current automated/manual Army fuel system as the functional and procedural baseline for the development of the Fuel Automated Subsystem.

B. FORCE XXI

1. Changing Missions and the Future Force

For the United States Army, the decade of the 90's has been a period of dramatic change. Change has not only occurred in terms of restructuring the force and increasing worldwide operations, but also in the doctrine in which military operations are conducted.

Change in combat operations first became apparent in combat operations at the end of the Cold War. Those combat operations included "Just Cause" in Panama in 1989, "Desert Shield / Storm" in Southwest Asia in 1990-1991, and "Provide Comfort" in Southwest Asia which began in 1991. In these operations much of the change was derived from information-age capabilities, increased integration of sister service components into an effective battle team, more lethal, survivable, agile weapons systems, and more capable soldiers and leaders. Most recently, the Army's operational missions and experiences have been focused on peace enforcement. Peace enforcement operations are not new missions for the military, but have become more frequent in the unstable post Cold War world. Operations such as "Restore Hope", "Uphold Democracy", "Able Sentry" and "Joint Endeavor", all peace enforcement operations, have caused the Army to redirect materiel, funds, and personnel resources to accomplish these missions. These operations serve to illustrate the doctrinal shift from the forward-deployed Cold War defensive Army to a force projection Army. The lessons learned from these peacekeeping missions are that they are fluid and dynamic operations. Specifically, the Army has determined that peace enforcement requires comprehensive battlefield visualization, robust situational awareness, continuous planning and capability for non-contiguous operations [Ref. 3]. In order to meet the current changes and challenges, as well as those of the 21^{st} century, the Army undertook the mission to determine the force design, capabilities and doctrine best suited for the future in order to maintain a qualitative edge over potential adversaries.

2. Evolution of Doctrine and Force Structure

Historically, the United States Army's structure and doctrine has continuously evolved and improved. The Army has primarily changed due to the incorporation of results of operational experiences, new operational concepts, and experimentation in force design. The Center for Army Lessons Learned at Fort Leavenworth, Kansas has captured and collected information learned from operations and training experiences with the sole purpose of providing those lessons to improve force design, doctrine, training and organization. For example, the Center for Army Lessons Learned most recently has captured the operational changes and experiences of the Army's humanitarian assistance in Rwanda and Somalia, and peacekeeping operations in Somalia, Haiti and Bosnia. New and innovative operational concepts have also been continuously developed. In the past, new concepts such as the Air Land Battle and Active Defense have ultimately become adopted as cornerstones of the Army's warfighting doctrine.

The Army has also improved through experimentation - Advanced Warfighting Experiments (AWE). Through virtual and live simulations, new approaches to land combat can be tested. Virtual simulation enables force developers to conduct progressive iterations that ultimately move closer to the ideal force. Live simulations and exercises insure real soldiers and units can ultimately execute their required warfighting operations. Today, through the approaches of operational experience, operational concepts, and experimentation, the Army has developed a new operational concept for land warfare - Force XXI.

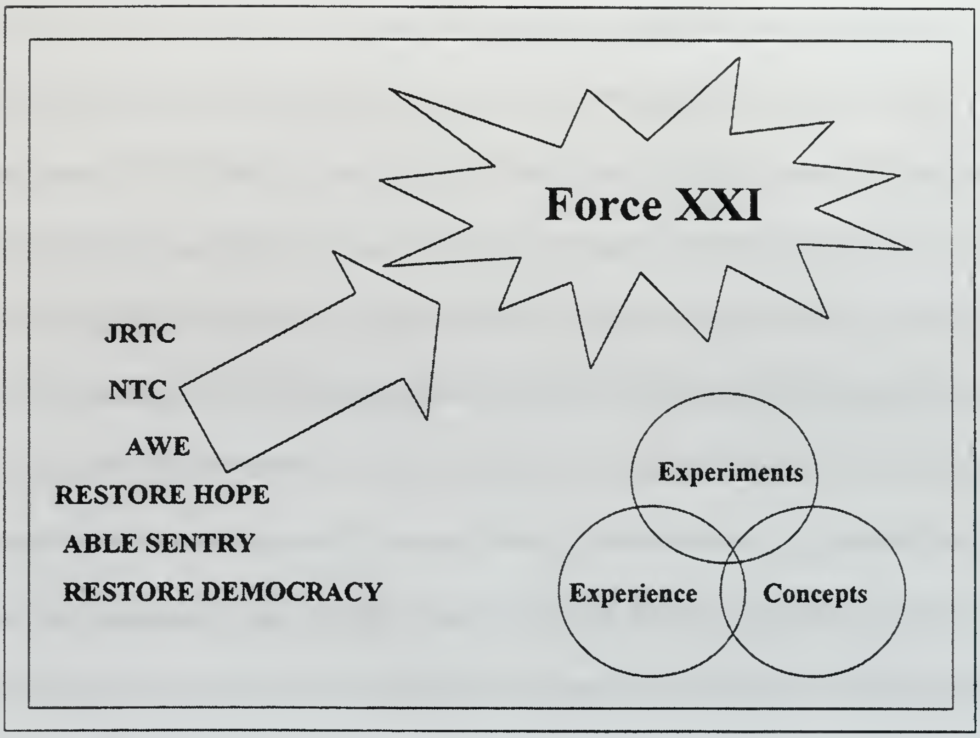


Figure 1. Evolution of Force XXI

3. Characteristics of Force XXI Operations

Force XXI, the United States Army of the 21st century, is the re-conceptualization and redesign of the Army's fighting force at all echelons, from the foxhole in the field to the industrial base, to meet the needs of a volatile and ever changing world. Force XXI operations are defined as multi-dimensional, precise, non-linear, distributed, simultaneous, and integrated [Ref. 3].

<u>*Multi-Dimensional*</u> – operates in a battlespace that maintains the traditional physical dimensions of width, depth, and height as well as the non-traditional dimension of the electro-magnetic spectrum.

<u>*Precise*</u> – characterized by synchronized attacks on vulnerable units and targets. These synchronized operations will require great precision. Precision is enabled by three capabilities. First, digitization of information allows informed decisions at all levels.

Second, sensors at the tactical, operational, and strategic level linked to analytical teams provide situational awareness. Lastly, simulations enable forces to be tailored to best meet an emerging situation or crisis.

Non-Linear – tasks executed across the entire battlespace rather than the massing of combat power at the forward line of troops. Non-linearity will also increase the requirement for all Army units (maneuver, combat service, and combat service support) to provide enhanced security.

Distributed Operations - enables units to take advantage of inter-networked communications and eliminates the use of the chain of command as the sole source of information. Decentralization allows subordinates to operate independently within the commander's intent and ultimately allows greater flexibility to changing situations.

Simultaneous Operations – mult-dimensional, precise, distributed and non-linear operations result in the ability of forces to conduct simultaneous operations across the entire battlespace. Digitization creates the ability to plan, coordinate, and execute operations simultaneously.

Integration – fully integrated with joint and allied forces. From initial mission receipt, deployment, mission execution, and transition to follow on operations, Army units will function as part of a joint task force.

4. Leveraging Information Technology

Force XXI is to be largely organized and focused around information and information technologies. The central and essential feature of this future Army will be its ability to exploit information. Two of the main objectives of Force XXI that allow future exploitation are the digitization of the battle-space and situational awareness.

Digitization of the battle-space simply refers to the application of technology to acquire, exchange, and employ timely information that is horizontally and vertically integrated to create a common picture of the battle field for soldiers and commanders [Ref. 3]. Situational awareness provides the capabilities of accurate and real-time information of friendly, enemy, neutral, and noncombatant locations. Additionally, situational awareness provides a common, relevant picture of the battlefield scaled to a specific level of interest and need [Ref. 3]. A digitized battle-space, coupled with situational awareness, will ultimately create a synergy among organizations that will greatly enhance the Army's combat capabilities and power projection.

C. FORCE XXI COMBAT SERVICE SUPPORT

1. Combat Service Support Defined

Combat service support (CSS) and the term logistics are closely related but yet are very different. Joint Chief of Staff Publication 4.0 defines the broad area of logistics as:

> "the science of planning and carrying out of the movement and maintenance of forces in its most complete sense: those aspects of military operations which deal with (a) design and development, acquisition, storage, movement, distribution, maintenance and disposition of materiel; (b) movement, evacuation, and hospitalization to include all levels of treatment of personnel; (c) acquisition or construction, maintenance, operation, and disposition of facilities; and (d) acquisition or furnishing of services" [Ref. 4].

Combat service support includes logistics in the battlefield functional areas of manning, arming, fixing, fueling, moving, and sustaining soldiers and their systems. Combat service support is characterized by anticipation, integration of functions, continuity of support responsiveness, versatility to circumstances, and improvisation.

2. Logistical Objectives

To meet the demanding challenges of Force XXI, providers of combat service support will be forced to leverage current and emerging technologies. Force XXI requires that combat service support be provided faster, more effectively and efficiently than ever before. To meet these requirements, the future strategic vision of "seamless support" was jointly developed by the Deputy Chief of Staff for Logistics (DCSLOG), United States Army Materiel Command (AMC), and the Combined Arms Support Command (CASCOM). In order to create a seamless continuum for logistics, the commanders from AMC, DCSLOG, and CASCOM approved The United States Army Strategic Logistics Plan (ASLP) that documents the strategy, initiatives and organizational structure to design and implement the logistical support system for Force XXI. The Army's Chief of Staff, General Gordon R. Sullivan, has called the ASLP, the "roadmap to take logistics into the 21st century" [Ref. 5]. The Army has defined eight objectives in the Strategic Logistics plan that will serve as the means to achieve the vision of "seamless support". Specifically, the eight Army strategic logistical objectives are [Ref. 2]:

(1) Develop a single, seamless logistics system.

(2) Standardize operating practices, automation, and communications.

(3) Reengineer logistics functions.

(4) Establish visibility of stocks and an integrated distribution system.

(5) Design a flexible and modular logistics force structure.

(6) Develop effective mobilization, strategic deployment and redeployment, reconstitution, and joint logistics capabilities.

(7) Include technology insertion, technical enhancements, and digitization.

(8) Establish performance measures based on tactical requirements and standards.

Using the objectives above, CASCOM is responsible for creating the Force XXI logistical organizations capable of "seamless support." To insure the support required by Force XXI is provided, future combat service support organizations must simply be designed to operate at an equivalent or higher level than the combat forces they support. The faster and more responsive combat support structure will require greater control of resources, information and communications. As a result, the logistics community can no longer rely on large inventories managed by telephone and the legacy systems of the past. In order to meet these strategic logistics objectives, the logistics community has undertaken several initiatives that develop information-based systems to support the future combat force. One of these main initiatives is the development of the Integrated Combat Support System (ICS3).

D. INTEGRATED COMBAT SERVICE SUPPORT SYSTEM

1. Current CSS Structure (Pre-Integrated Combat Service Support System)

a. Procedures and Operating Methods

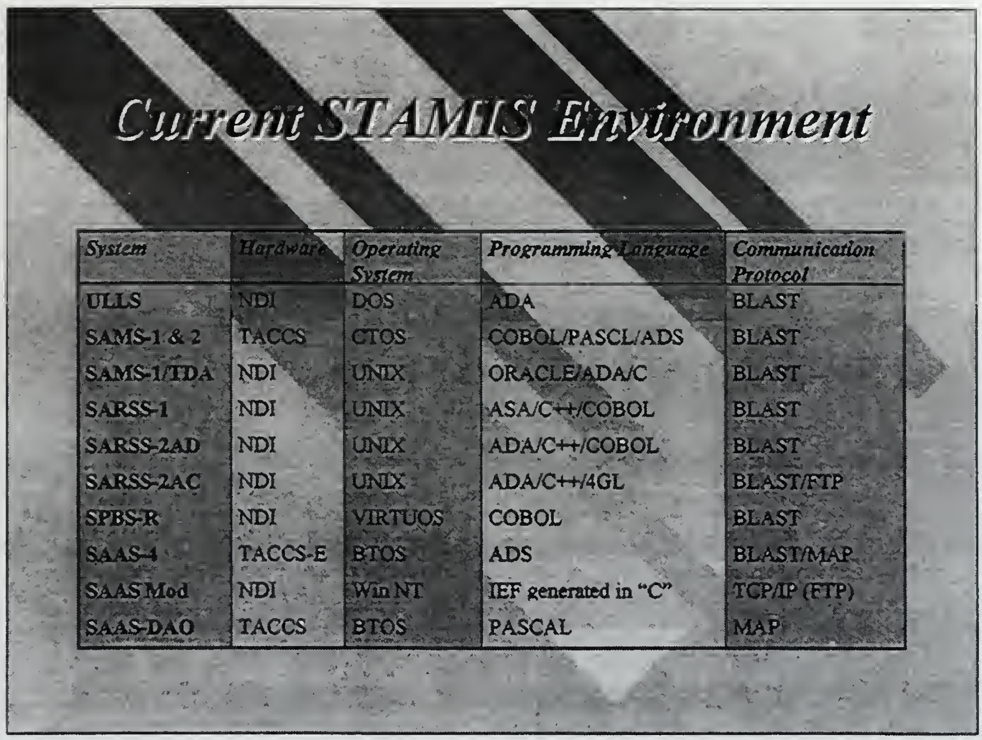
The current policies and procedures for accomplishing combat service support functions are directed by various Department of the Army and Department of Defense publications (i.e. MILSTRIP and MILSTAMP). These directed data formats and procedures are used by all organizations and systems that deal the management of personnel, finance, logistics, and distribution of materiel from the acquisition of the item to its ultimate consumption/use by the end user. Today's automation procedures have many shortfalls and use a combination of batch processing, mini-batch, and interactive

processing. Many of these automated systems work separately with limited data sharing and source data is almost non-existent. Currently, there is an insufficient level of automation and corresponding communication networks to support the CONUS-based force projection strategy described in FM100-5, Operations. Additionally, there is no universal theater/national asset visibility or management tool available for the logistician and the commander to assess materiel and readiness posture.

b. Information Flow

Current logistical information flow is both manual and automated in nature. Data is physically transported vertically to and from the lowest tactical level to the national level. Accordingly, this vertical information flow of manual data hinders the corresponding horizontal flow. The manual data flow process is characteristically a time and labor intensive process which is prone to error, duplication, delays, and loss of data. Manual methods are used at all Army levels until eventually entry points to an automated system are reached. To minimize the amount of manual information flow, automation has been progressively implemented throughout the Army.

Individual Army combat service support automation systems, known as Standard Army Management Information Systems (STAMIS) have developed separately during the evolution of the information age. However, data cannot be efficiently transferred vertically or horizontally within a functional system, or between such systems, due to their "stove-pipe" nature. Information flow is routinely hindered by antiquated, obsolete, and un-linkable computer systems, software languages, operating systems, varying source data input, and by the lack of connectivity/interfaces.



System	Hardware	Operating System	Programming Language	Communication Protocol
ULLS	NDI	DOS	ADA	BLAST
SAMS-1 & 2	TACCS	CTOS	COBOL/PASCL/ADS	BLAST
SAMS-1/TDA	NDI	UNIX	ORACLE/ADA/C	BLAST
SARSS-1	NDI	UNIX	ASA/C++/COBOL	BLAST
SARSS-2AD	NDI	UNIX	ADA/C++/COBOL	BLAST
SARSS-2AC	NDI	UNIX	ADA/C++/4GL	BLAST/FTP
SPBS-R	NDI	VIRTUOS	COBOL	BLAST
SAAS-4	TACCS-E	BTOS	ADS	BLAST/MAP
SAAS Mod	NDI	Win NT	IEF generated in "C"	TCP/IP (FTP)
SAAS-DAO	TACCS	BTOS	PASCAL	MAP

Figure 2. Current "Stove-pipe" STAMIS [Ref. 6]

This lack of information sharing and efficient system interfaces can lead to supply requests which may not be processed in a timely and accurate manner. Users of the current automated systems often must hand-carry data (magnetic media and disks) across the battlefield to perform routine combat service support missions. This messenger network is commonly referred to as the “sneaker net”. In some cases, data must be retrieved from one functional system, downloaded, reformatted, manipulated, and then reentered into another related functional system that requires the same exact data elements and information, thus causing delays and errors. Further impeding information flow, the financial, personnel, and unit movement automation has not been fully integrated with other combat service support systems.

As a result of fractured combat service support automation, customers have developed a number of unique systems that do not work with the standard systems

or support combined/joint automation requirements. In short, for combat service support automation, presently there is no seamless and interactive flow of information but rather distinct islands of automation.

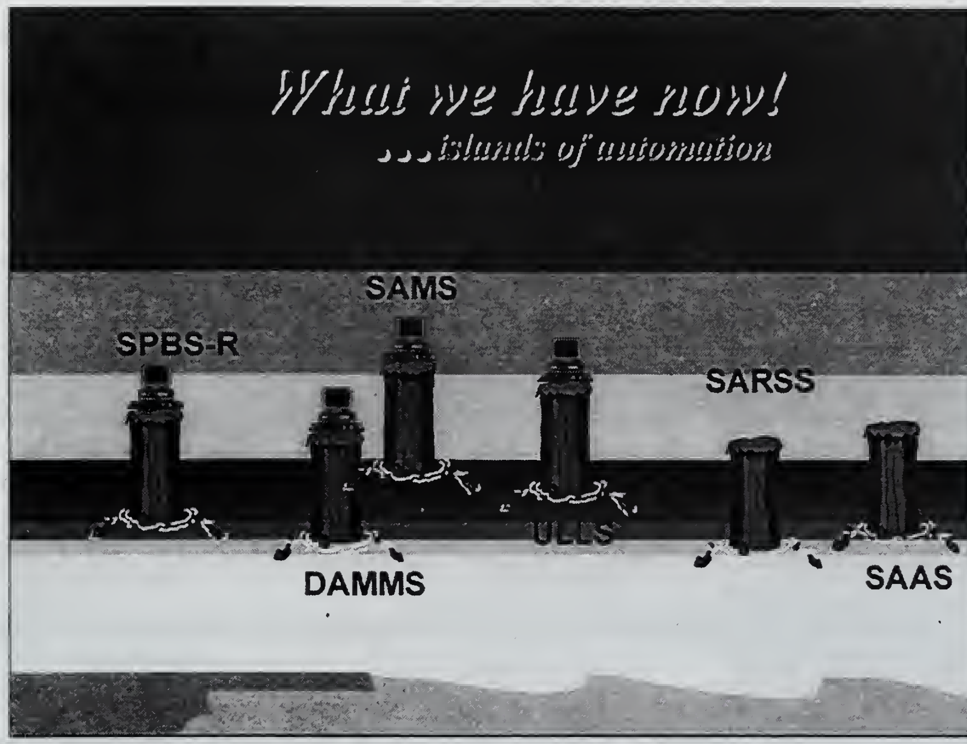


Figure 3. Current STAMIS Environment [Ref. 6]

c. Security

Army combat service support automation security is established in accordance with AR 380-19, Department of the Army Information Security Program. A password and user identification that limits system access is commonly used, and controlled by the system administrator. The password/user identification system coupled with physically separate systems serves as the primary means of maintaining security for combat service support systems operating at different security levels. This total physical separation between automated systems at different security levels also contributes to

interrupted data flow. Output reports from automated systems are required to contain sensitivity markings, and audit trails are maintained internally by the systems.

2. Integrated Combat Service Support System Overview

To meet the shortfalls in current Army combat service support systems, the Integrated Combat Service Support System (ICS3) will be the total Army's (i.e. active Army, Army Reserve, and National Guard) single seamless, integrated, and interactive combat service support management system [Ref. 7]. ICS3 specifically supports the Force XXI digitized Army and will also constitute the Army's portion of the Global Combat Support System (GCSS). ICS3 also will enhance situational awareness and encompass all combat service support disciplines so as to man, arm, fix, fuel, move, and sustain the force under one single automated system. ICS3, as a single automated system, will maximize the use of source data automation to reduce data transcription errors and improve accuracy. The structure of ICS3 will be modular and will be able to function in support of split based operations, joint operations, and operations supporting allied nations.

3. Development of the Integrated Combat Service Support System

While ICS3 supports situational awareness for command and control, its primary purpose is to automate the combat service support processes for the total Army. The ICS3 will be seamless, integrated, modular, and interactive at all force support levels. The system will operate on commercial off the shelf (COTS) non-developmental item (NDI) computer equipment. The modular design of ICS3 allows the system to be tailored to accommodate different combat service support missions and organizations. ICS3 will

further provide combat service support units at all levels, a responsive and efficient capability to anticipate, allocate, and synchronize resources, services, and information.

Development of ICS3 will not be a totally new program. The development of ICS3 is scheduled to follow a three phased evolutionary strategy.

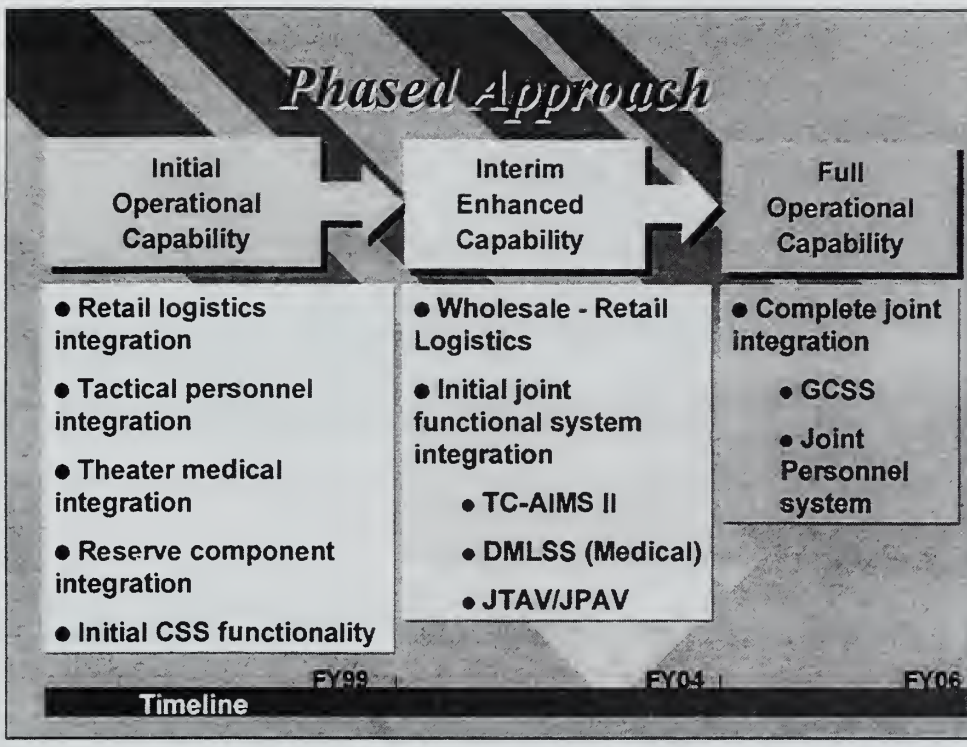


Figure 4. ICS3 Phased Development [Ref. 6]

Phase I provides an initial operational capability. Phase II provides an interim enhanced capability, and Phase III is scheduled to ultimately achieve full operational capability. The system development will initially integrate and modernize the current logistics STAMIS, then implement improvements that integrate wholesale and retail combat service support, and eventually achieve joint integration.

<u>*Phase I – Initial Operational Capability:*</u> In this phase, an initial operational capability will be developed through integration and modernization of the current tactical

logistics STAMIS. The capabilities to be integrated during this phase are supply, property, ammunition, and maintenance functions. The corresponding tactical logistics STAMIS to be functionally integrated include: the Unit Level Logistics system (ULLS); Standard Army Retail Supply System (SARRS); Standard Army Property Book System (SPBS); Standard Army Ammunition System (SAAS); and the Standard Army Maintenance System (SAMS). The planned modules include:

(1) a supply and property module that integrates supply operations and property accountability.

(2) a modernized maintenance module that integrates maintenance operations (ground, aviation, petroleum equipment, etc.) at each level of maintenance.

(3) an ammunition supply point module that integrates class V management and operations at ammunition supply points.

(4) a supply support activity module that integrates supply management and operations at supply support activities.

(5) an integrated materiel management module that integrates supply, property, ammunition, and maintenance management in all materiel management organizations.

(6) A management module that integrates information from multi-functional CSS data sources and allows for data exchange with other ICS3 modules and external systems.

The ICS3 will also have interfaces so that users can gain access to information and exchange operational data in CSS functional areas such as personnel, medical, finance, transportation, training, and unit administration.

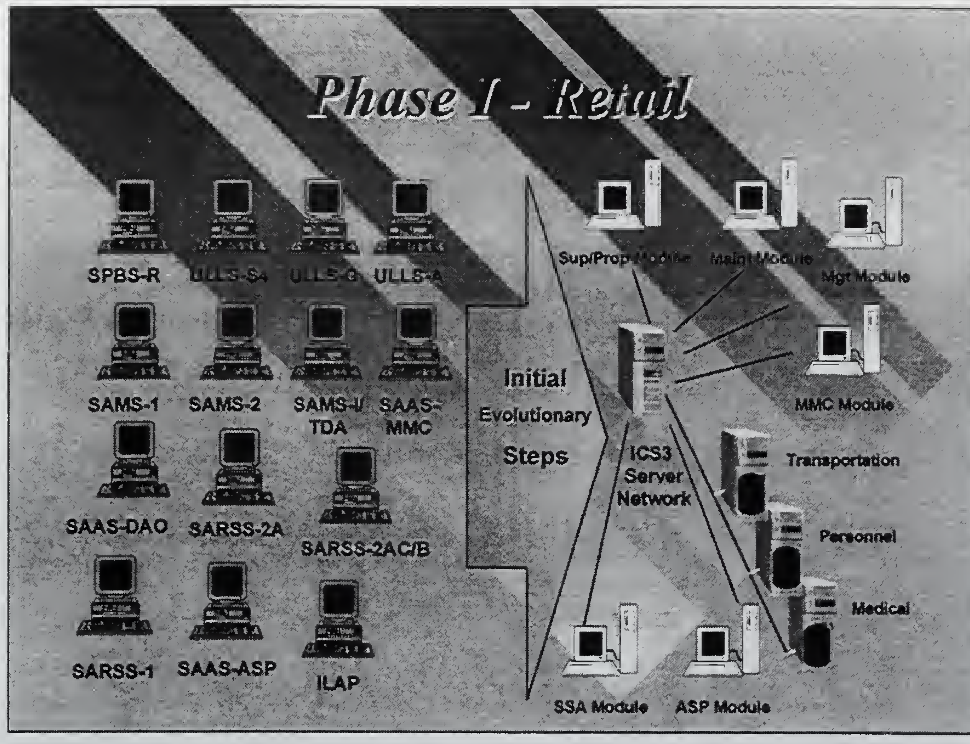


Figure 5. ICS3 Phase I [Ref. 6]

Phase II – Enhanced Operational Capability: In this phase, the initial ICS3 Phase I capabilities will be enhanced by including the processes associated with wholesale logistics. The design and development of ICS3 during this phase will be shaped by further advances in technology, results from advanced warfighting experiments (AWE), new battlefield distribution concepts and Force XXI initiatives.

Phase III – Full Operational Capability: This phase will be completed with the implementation of all required interfaces to the automation systems of the joint community and applicable allied systems. The full operational capability will provide the seamless, integrated, modular, interactive, and interoperable combat service support system for the total Army.

4. Integrated Combat Service Support System End State Vision

The final state of the ICS3 developmental plan will ultimately result in a "system of systems" that automates the combat service support functions of manning, arming, fixing, fueling, moving, and sustaining the force. The ICS3 will have unified, organized and enhanced the legacy STAMIS initiatives of both retail and wholesale logistics. The system will provide one common integrated source of logistical data.

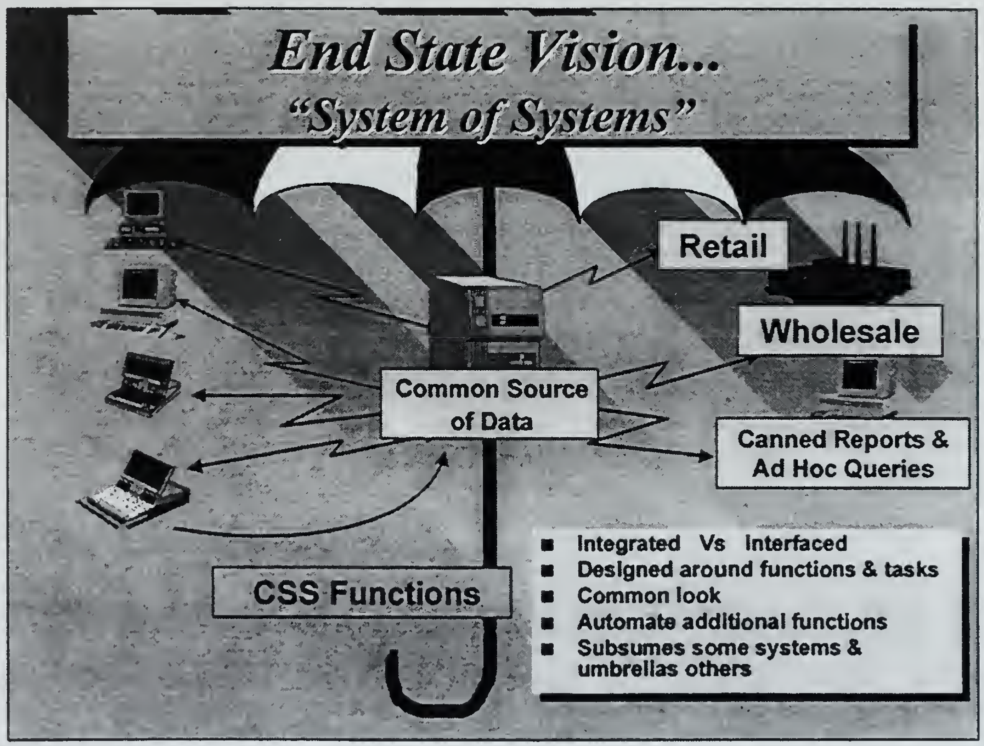


Figure 6. ICS3 a "System of Systems" [Ref. 6]

The benefits of ICS3 are great and can most clearly be illustrated in the simple scenario of a unit commander who would like to determine the status of the unit's motor pool. Currently, the unit commander must extract information from six different STAMIS that are most likely not co-located in order to obtain an overall picture of the status of his unit's vehicle fleet.

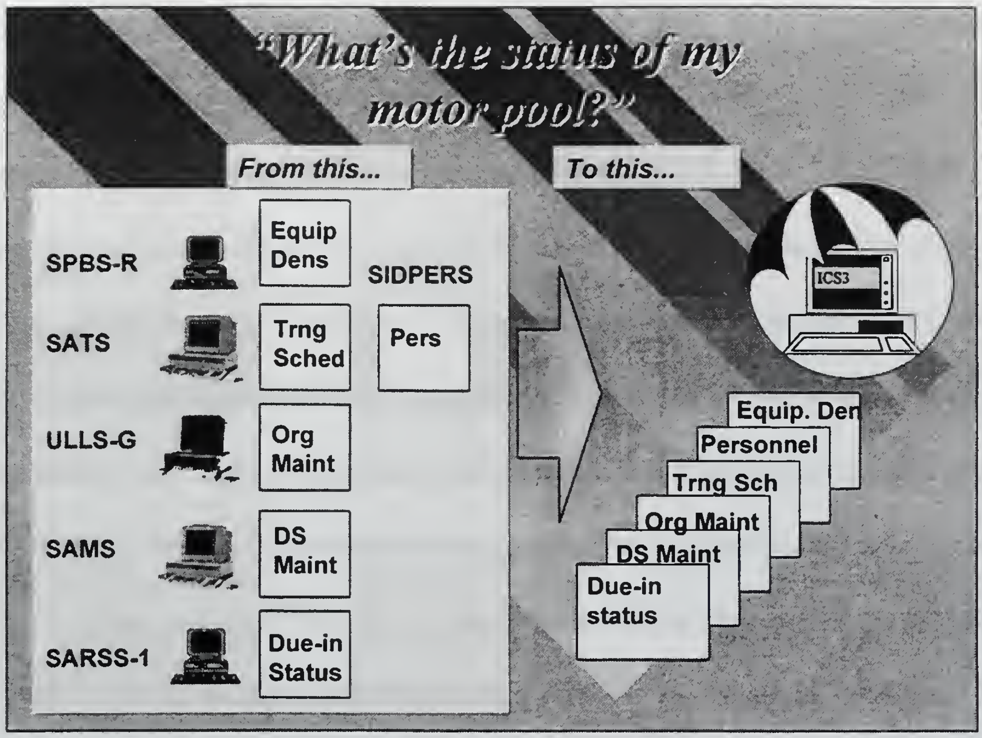


Figure 7. Motor pool example of the benefits of ICS3 [Ref. 6]

From the Standard Property Book system (SPBS-R) located at the property book office, the commander can determine the unit's equipment density; the number of vehicles currently on-hand verses the number authorized. From the Standard Army Training System (SATS) in the S-3 office and Standard Army Installation/Division personnel System (SIDPERS) in the S-1 office, training and personnel strength can be determined. To determine which vehicles are non-mission capable, the Unit-Level Logistics System (ULLS-G) and Standard Army Maintenance System (SAMS) located in the motor pool must be accessed. Lastly, the Standard Army Retail Supply System (SARRS) displays the due-in status for the parts needed to fix any non-operational vehicle. In this scenario, there is no STAMIS that provides the commander or logistician an accurate assessment of the unit's fuel status. Under ICS3, the commander will simply

use one computer to navigate through a series of icons/windows to have complete visibility of the motor pool.

On a larger scale, ICS3 supported by a responsive communications network can enhance the battlefield commander's decision making capability by providing timely and accurate status reports on units, equipment, and supplies. As a result, the commander will have the capability to optimize allocation and use of limited CSS assets to meet the needs of the force. By having a common integrated automation system, the collection, processing, and transferring of unnecessary or redundant data from the tactical, strategic, and operational levels will be eliminated.

E. CURRENT ARMY FUEL SYSTEM PROCEDURES & AUTOMATION

Phase I of the ICS3 developmental plan develops initial operational capability to include retail logistics through the integration and modernization of the current tactical logistical STAMIS. At the retail level, the current Army fuel system is only partially automated. No dedicated STAMIS exists for fuel management and most procedures are still conducted manually. As a result of limited automation, the Army's fuel supply system has distinct shortcomings. These shortcomings were identified during the FY 84 through FY 91 U.S. Army Audit Agency audits of 27 Army installations. The audit of bulk petroleum management and accountability yielded three main conclusions [Ref. 8]:

(1) Supply Support Activities and using units did not adequately account for bulk petroleum products.

(2) Control of deliveries from contractors was not effective.

(3) Physical security measures did not adequately safeguard petroleum products

To alleviate the shortcomings during ICS3 development phase I, the current limited retail fuel automation capabilities must be integrated and the manual procedures modernized / automated when developing the Fuel Automated Subsystem.

1. Petroleum Product Classifications and Retail Operations

The term's fuel and petroleum are synonymous with class III. Class III is the common Army supply classification for petroleum products, and includes bulk fuels (i.e. diesel, gasoline, etc.) and packaged petroleum products (i.e. motor oil, grease, antifreeze etc.). Packaged petroleum products are managed/accounted for by the same procedures that apply to other classes of supply (i.e. repair parts, expendable supplies, etc.) and will be captured in the development of the Standard Supply Subsystem within the Supply Support Module of ICS3. Accordingly, the remaining sections only address current bulk fuel functions and management at the retail level since it serves as the basis for determining the requirements of the Fuel Automated Subsystem.

Retail fuel operations are conducted at class III supply points and include the general functions of receipt, storage, issue, and quality assurance / surveillance. However, inherent in these basic operations is the requirement to manage petroleum operations. Petroleum management involves maintaining accurate accounts and executing appropriate quality control procedures for all receipts, issues, and stocks on hand.

2. Manual Fuel Management

At the class III supply point, accountability is maintained through managing the receipt, storage, and issue of class III products. AR 710-2, Inventory Management Supply Policy Below the Wholesale Level, specifically mandates the requirement to

maintain an accurate account of all petroleum receipts, issues, and stocks on hand. To accomplish the inventory and accountability requirements of bulk fuels per Army regulation, the following five manual documents and reports are currently used:

DD Form 1348-1 – Whenever bulk petroleum is received at the Class III supply point, Department of Defense (DD) Form 1348-1 (DOD Single Line Item Release/Receipt Document) is generally used as the receipt document. The form is a six-part, carbon-interleaved form. The form is specifically used to verify that the supply point has received the correct amount and type of fuel. Upon receipt, an individual verifies the type and amount of fuel and then completes the form by signing and dating it.

DA Form 2765-1 – The supply point customer uses Department of the Army (DA) Form 2765-1 (Request for Issue or Turn-in) to request packaged and bulk products to be delivered by tank trucks and semi-trailers. The form may also be used for the turn-in of excess cans, drums, or supplies. The individual making the issue completes the form by annotating the issued quantity, initialing the form, and then dating it.

DA Form 3643 – The DA Form 3643 (Daily Issues of Petroleum Products) is a detailed daily record of all fuel products issued and received at the supply point.

DA Form 3644 – The DA Form 3644 (Monthly Abstract of Issues of Petroleum Products and Operating Supplies) is used to show the total monthly issues and receipts of petroleum products and operating supplies.

DA Form 4702-R – The DA Form 4702-R (Monthly Bulk Petroleum Accounting Summary) is used to report all losses or gains revealed by monthly inventories. Inventory losses reflected on this form which exceed those allowed by AR 703-1, or that are disapproved by the approving authority (the first lieutenant colonel in the chain of

command) must be supported by a report of survey. Gains in excess of the allowable limits must also be investigated to determine the cause. A copy of the investigation report must be attached to DA Form 4702-R as a supporting document.

An illustration of each of these forms is provided in Appendix A of this thesis. Additional guidance on the use of these forms can be found in Department of the Army Pamphlet 710-2-1 (Using the Unit Supply System: Manual procedures) and 710-2-2 (Supply Support Activity Supply System: Manual Procedures). A summary of the overall interaction between these fuel accountability documents as prescribed in AR 710-2 and DA Pam's 710-2-1- and 710-2-2 is illustrated below.

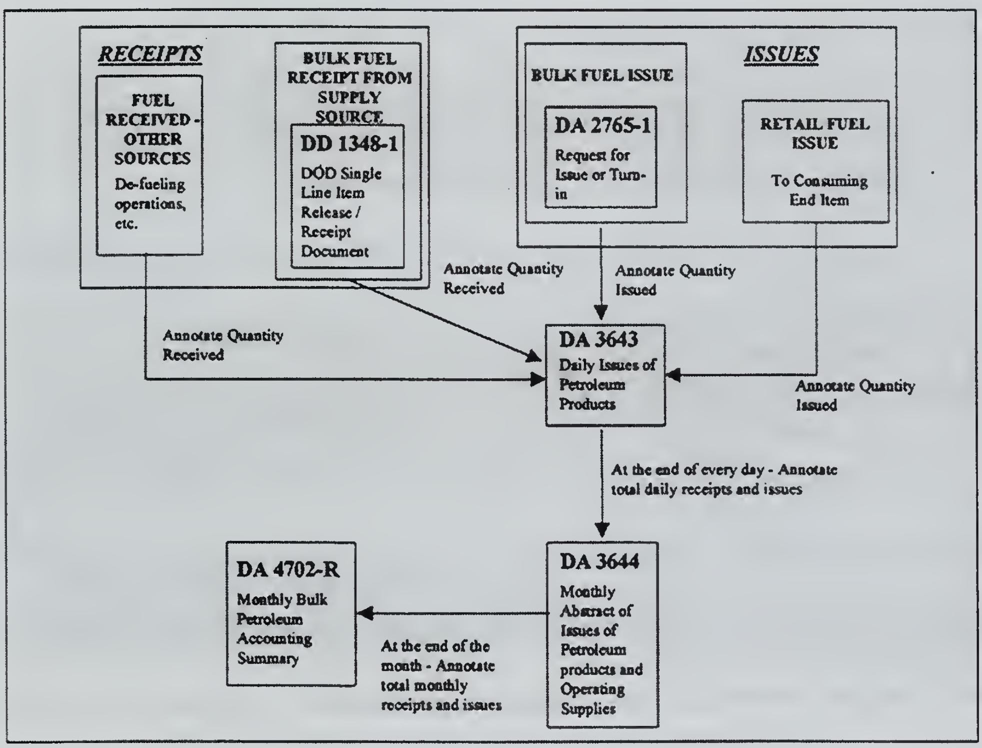


Figure 8. Fuel Documentation Flow

Also inherent in fuel management is the aspect of quality surveillance. The quality surveillance mission is to maintain the quality of petroleum products from point of origin to the point of use. The quality surveillance program encompasses bulk fuel in

waterborne carriers, tank cars, tank vehicles, pipeline systems, bulk storage, and packaged products. Detailed information regarding the specific procedures for inspecting, sampling, testing, and handling for each storage and/or transportation mode is contained in the references listed below.

Bulk Storage – AR 703-1, DOD 4140.25M, Federal Test Method Standard 791, Military Handbook 200, Military Handbook 201, Military Standard 140, Military Standard 140, Military Standard 161, and Military Standard 457.

Bulk Transportation Classes :

Marine – Commander of Military Sea-lift Command Instructions 3121.3, DOD 4140.25M, Military Handbook 200, and Military Handbook 201.

Tank Cars and Tank Vehicles – AR 703-1, DOD 4140.25M, FM 10-69, FM 10-71, Military handbook 200, and Military Handbook 201.

Pipeline – AR 703-1, AR 715-27, DOD 4140.25M, FM 10-18, FM 10-20, FM 10-70, FM 10-207, Military Handbook 200, Military Handbook 201, and Military standard 161.

In all cases, the procedures for inspecting, sampling, testing, and handling are accomplished manually. These tasks involve soldier involvement to actually take a sample of a petroleum product, conduct laboratory tests, etc. The physical procedures required in quality surveillance are currently not automated, and would prove difficult to automate.

3. Automated Fuel Management

Fuel management is automated to a very limited extent. Currently, fuel requisitions for both packaged and bulk petroleum products can be entered in the Army's

automated supply system through ULLS, the Unit Level Logistics System. From ULLS, the requisition transitions through SARSS and SAILS (the intermediate supply level), where it may be filled. If the requisition is not filled at the intermediate supply level, the requisition is passed to the Defense Fuel Supply Center's Fuel Automated System (the whole sale supply level). The Defense Fuel Supply Center (DFSC) acts as the Defense Logistics Agency's (DLA) designated representative, conducting all of the Department of Defense's management of fuel inventories and war reserves. A diagram of this procedure and the different system functionality is illustrated below.

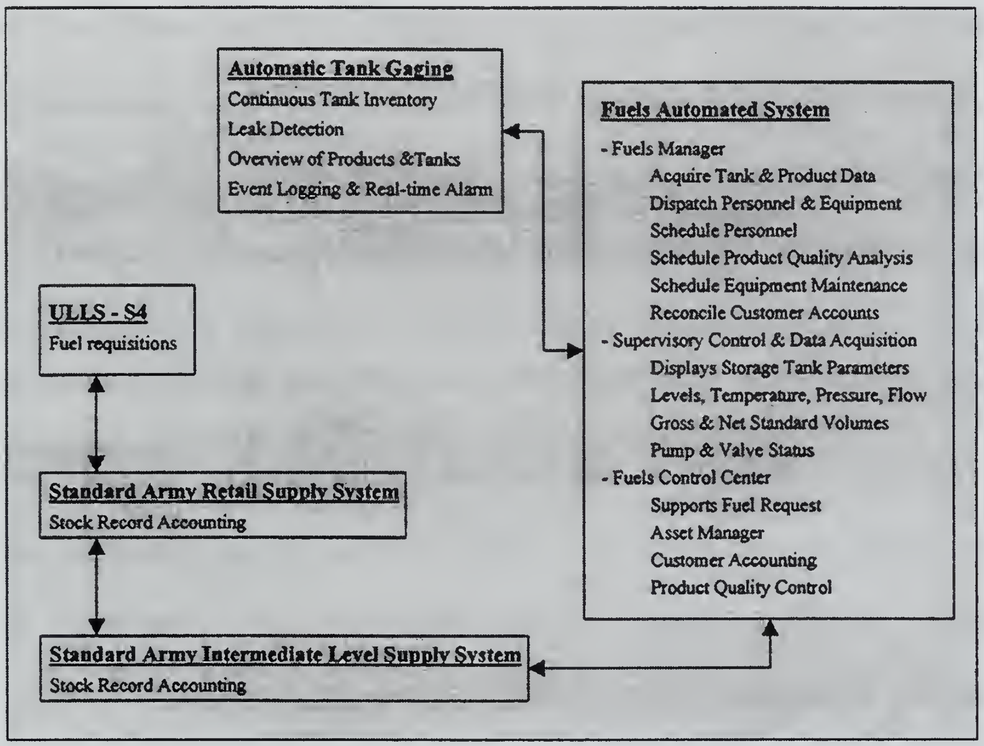


Figure 9. Current Fuel Automation Structure

The automated flow, however, only handles requests for petroleum products. All retail level management functions associated with the receipt, issue, and storage of bulk fuels are not automated.

A new initiative to provide fuel asset visibility has most recently been incorporated within the Army's Advanced Warfighting Experiments. Applique, is a new software/hardware suite that digitizes command and control at brigade level and below. Applique, is also a subsystem of Force XXI Battle Command, Brigade and Below (FBCB2) and provides the commander or logistician a relevant picture of the current combat service support situation at his/her echelon as well as visibility to subordinate units. It specifically is designed to provide combat service support situational awareness and enhanced capability to synchronize support to customers. The combat service support functionality in Applique includes logistics situational reports, personnel situational reports, situational awareness, requests for support, and logistical task order messaging.

Applique is designed to interface with another new test system - the Combat Service Support System (CSSCS). CSSCS provides desired command and control information to CSS and force level commanders' and their staffs based upon data received from the legacy CSS STAMIS and CSS functional elements.

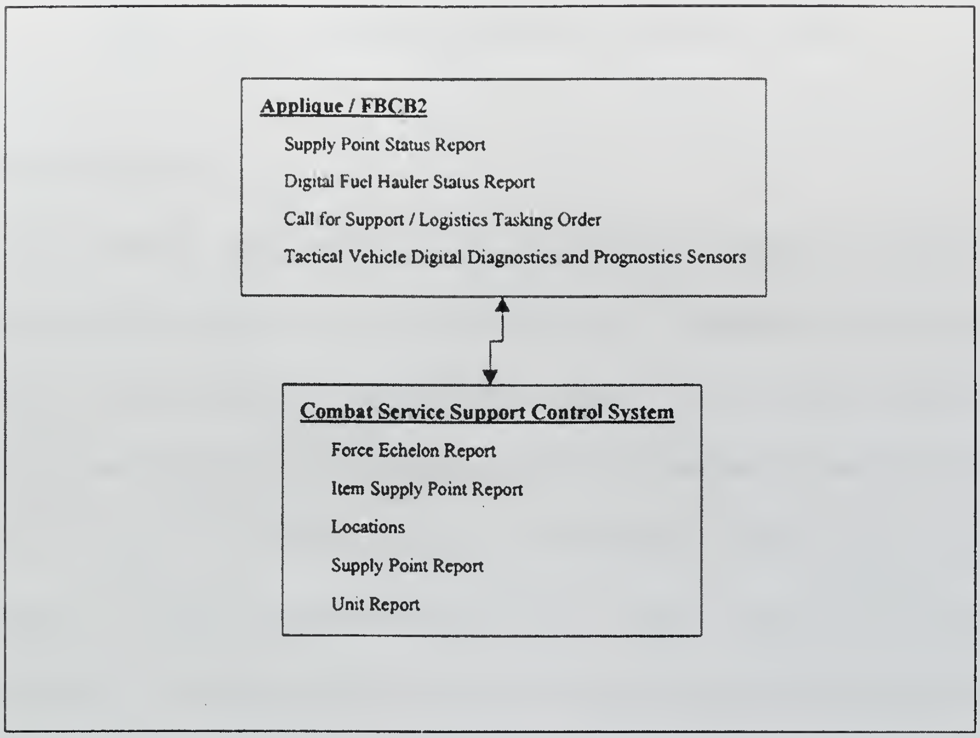


Figure 10. New Fuel Automation Initiatives

While these new test systems provide situational awareness and fuel asset visibility, retail fuel management functions associated with the receipt, issue, and storage of bulk fuels remain un-automated.

III. COMPUTER AIDED PROTOTYPING SYSTEM

A. INTRODUCTION

The Computer Aided Prototyping System (CAPS) is a computer aided software engineering tool, developed at the United States Naval Postgraduate School, used for automated, real-time software prototype development. In this thesis, CAPS is the software engineering tool used to develop a prototype of the Fuel Automated Subsystem of ICS3. The objective of CAPS is to "assist Department of Defense program managers and engineers to rapidly evaluate requirements for military real-time control software using executable prototypes and to test and integrate completed subsystems through evolutionary prototyping" [Ref. 9]. CAPS is a self-contained, complete developmental environment that incorporates graphical system decomposition and design, interface design and integration, real-time scheduling, and compiler support. CAPS also provides automated support for software reuse.

B. THE PROTOTYPING PROCESS

1. Traditional Software Development

Prototyping is widely used in conventional industrial manufacturing to refine requirements and to provide a tangible product that can be verified against customer needs. In software development, however, prototyping is not widely used.

The traditional approach to software development is commonly known as the "waterfall" method. In the "waterfall" method, a developmental project proceeds in distinct phases - analysis, design, coding, testing, and maintenance.

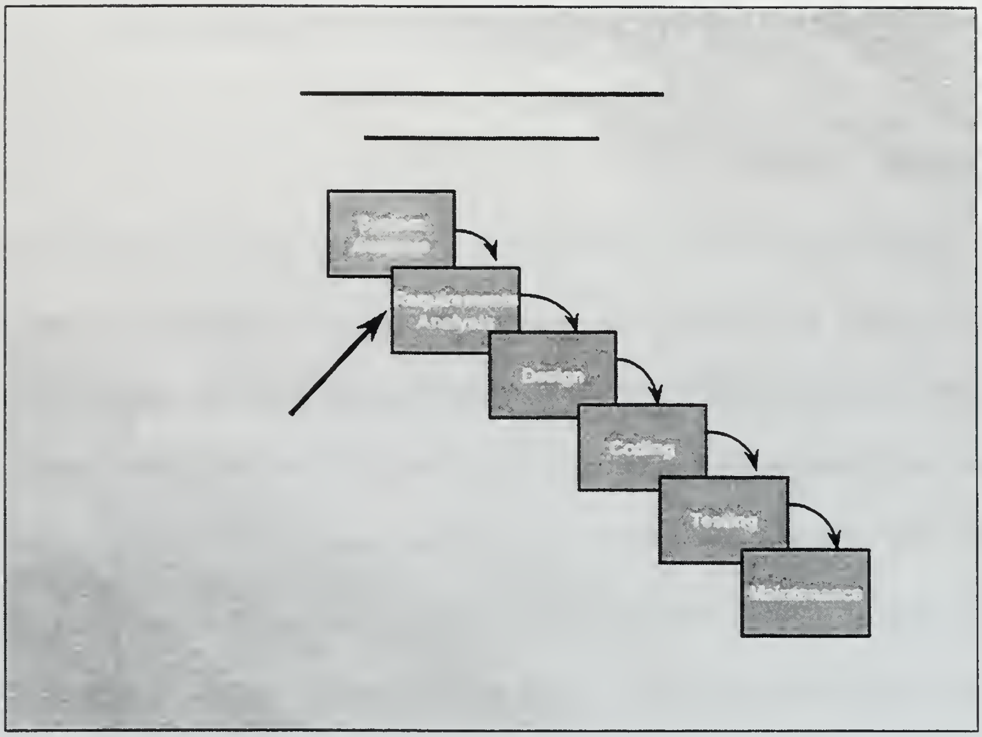


Figure 11. Traditional Software Development - "Waterfall Method" [Ref. 10]

With the "waterfall" methodology, the testing phase at the conclusion of the project is an extensive attempt to insure the system functions properly and meets customer needs. However, the major flaw in this approach is that system requirements/functionality is determined early during the requirements analysis phase. Requirements can not be simply frozen during system development, but rather tend to change and evolve. Additionally, without trial use of the system, the customer cannot effectively validate requirements. As a direct result of these distinct problems, requirement errors are only discovered at the end of development in the maintenance phase, when typically the budget is nearly expended and the target system development date has past. The ultimate consequence is that there are no immediate ways to recover from any major errors without a significant investment of time and money. The cost of a requirement error rises exponentially during the software developmental process.

Specifically, discovering a requirement error at the end of the "water fall" developmental process costs approximately 100 times more to fix than if it was initially identified early during the requirements analysis phase [Ref. 11].

2. Rapid Prototyping

Prototyping is the process of developing a "concrete executable model of selected aspects of a proposed system" [Ref. 12]. Rapid prototyping is a further extension of the prototyping that allows the quick development of an executable program that demonstrates feasibility, can be reviewed by the customer for accuracy, and easily modified/refined. The prototyping process is iterative in nature.

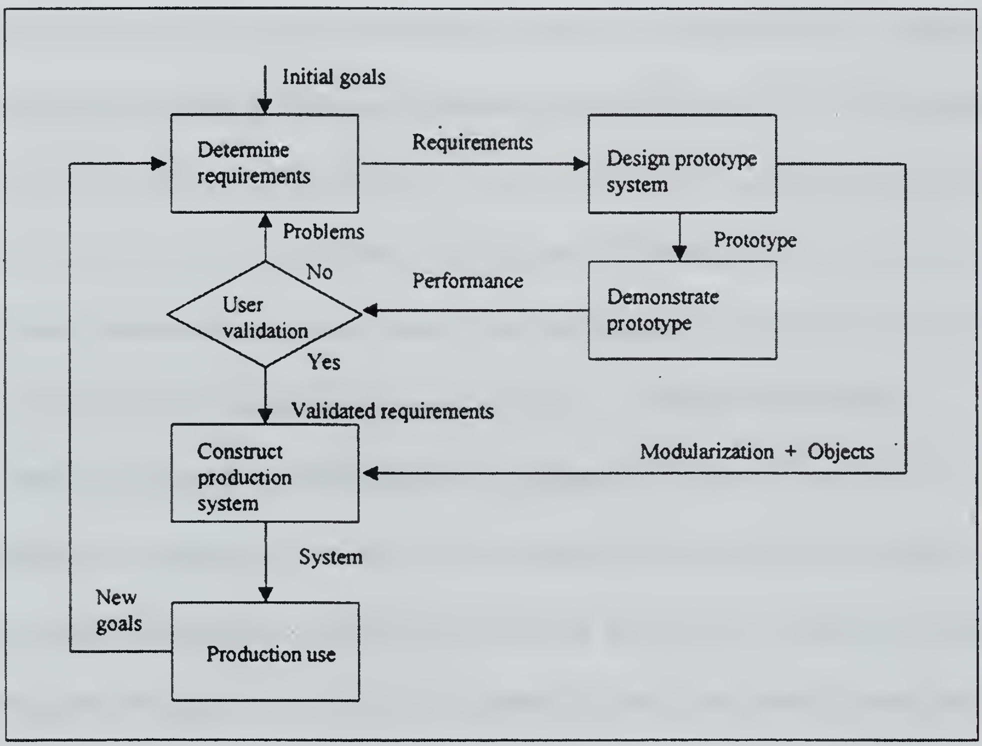


Figure 12. The Prototyping Cycle [Ref. 12, 13]

In rapid prototyping, the system designer and the user jointly develop initial requirements and specifications for the critical parts of a system. These requirements serve as the basis to design a prototype that can be demonstrated to the customer. The

prototype is a partial representation of the system and includes only the necessary attributes to meet the requirements. The prototype simply serves as an aid in analysis and design. From the prototype, the customer can clearly determine if the prototype meets his or her needs. If the prototype is accurate, development of a production system can proceed. However, if the prototype does not meet the customer's requirements, any shortcomings can be addressed and the requirements used to build the prototype modified. The prototyping cycle continues until the customer ultimately is provided a prototype that meets his/her needs. Once the valid prototype has been identified, the requirements captured in the prototype serve as the starting point for production code development. In some cases, the prototype itself may actually serve as the basis of the production code. Further system development is typically required since the prototype [Ref. 12]:

(1) may not include all of the aspects of the total system.

(2) may have been implemented using other resources not available in the actual system environment.

(3) may not be capable of handling the full demands of the actual system.

The prototyping approach solves the flaw of late identification of requirement errors in the traditional "water fall" software methodology. In the prototyping approach, the requirement changes are identified early during the iterative prototyping process as opposed to the maintenance phase of production code in traditional development. Since the prototype description is typically simpler than production code, it is more suited to easy modification at great cost and time savings. This flexibility also makes prototyping quite attractive for systems that have undefined or rapidly changing requirements.

C. CAPS DEVELOPMENTAL ENVIRONMENT

The Computer Aided Prototyping System (CAPS) is an integrated software development environment that is at the heart of the requirement generation process used in this thesis. The Prototype System Description Language (PSDL) in CAPS is designed for specifying real-time systems. The PSDL descriptions produced by CAPS provide a formal and unambiguous definition of the modeled system.

CAPS consists of four major components: a set of editors for design entry, a software base of reusable components, and an execution support system to construct the executable prototype. By using the CAPS graphics and text editors, the user can create a prototype that specifies the essential requirements of the system. The editor enforces consistency and enables rapid construction of an accurate model. The system uses the Transportable Application Environment (TAE) to construct a graphical user interface (GUI) for the prototype. Once the prototype is specified, the user translates and schedules the prototype and then automatically creates an executable driver program that incorporates the requirements of the specification file.

1. Prototype System Description Language

A prototyping system description language (PSDL) must be easy to read in order to serve as design documentation, but also must be formal enough for mechanical processing in the rapid prototyping environment [Ref. 14]. The Computer Aided Prototyping System's PSDL computational model is based on the following augmented graph [Ref. 14]:

$$G = (V, E, T(v), C(v))$$

The vertices, V, represent operators. The edges, E, represent data streams. The timing and control constraints T (v), C (v) can be applied to each vertex. The major constructs and concepts of the PSDL are operators, streams, types, timing constraints, and control constraints [Ref. 11].

a. Operators

Operators are designated in the CAPS graphical editor as either circles or rectangles. Circles represent proposed software components and the rectangles represent simulated external systems. Each operator is identified through a unique name and can also be assigned a maximum execution time. Operators can be further decomposed and are identified as double circles. Decomposed operators are known as composite operators. Any operator that is not decomposed is known as an atomic operator and will be ultimately implemented in the Ada programming language. Operators that output a value based solely on a set of input values are known as functions. Operators that have output based on one or more state variables are known as state machines.

b. Streams

Streams represent communications linking a set of one or more producer operators to a set of one or more consumer operators. There are two possible types of streams. The first type, the sampled stream, acts as the equivalent of a programming variable. The sampled stream models continuous data such as sensor output and only the most recent data value is of interest. The second type, the data flow stream is a stream that has a consuming operator triggered on every occurrence of data on the stream. The consuming operator removes each data occurrence after it has been read. Data flow streams model discrete transactions.

c. Types

PSDL contains standard predefined types such as boolean, character, string, integer, real, and type constructors (set, sequence, map, tuple, etc.). Additionally, the PSDL has the ability to handle user defined abstract data types.

d. Timing Constraints

Timing is a critical issue in modeling real-time software systems. Timers serve as a software stopwatch and are declared in the implementation portion of a composite operator. Any operator that is given a timing constraint is considered time critical, and is given scripted, static scheduling priority. There are two kinds of time critical operators - periodic and sporadic. Periodic operators are assigned a maximum execution time and a period. Additionally, a periodic operator may be assigned a finish within time. The maximum execution time is the scheduled processor time for execution of the operator. The period controls how frequently the operator executes its respective code. The finish within time forces completion of the execution during a portion of the period. Sporadic operators are assigned a minimum calling period as well as maximum response time. The minimum calling period places a lower bound on the time between consecutive input data arrivals. The maximum response time places an upper bound on the time between input data arrival and the completion of execution.

e. Control Constraints

Control constraints provide conditional execution of operators. Triggers and execution guards are used within the PSDL to gain conditional execution. The PSDL allows for two different types of triggers to restrict the conditions that an operator executes. The “by all” trigger is assigned to an operator when the user desires to restrict

the operator to executing only if all of the listed streams have new data. The "by some" trigger is closely related to the "by all" trigger, except that new data is required on only one stream to stimulate execution of the designated operator. Execution guards are conditional statements similar to an "if-then-else" statement, and are evaluated based on data received from one or more of the streams or state variables. If the execution guard condition is satisfied, the operator executes. If the execution guard is not satisfied, the operator simply does not execute.

2. Additional CAPS Resources

CAPS version 1.2 was used for the development of the Fuel Automated Subsystem. CAPS continues to be further enhanced at the United States Naval Postgraduate School. Additional information regarding CAPS, to include user manuals, prototype demonstrations and current CAPS projects, can be found on the Internet at [http://wwwcaps.cs.nps.navy.mil].

IV. REQUIREMENTS ANALYSIS AND MODELING

A. INTRODUCTION

Various models have been developed to describe the traditional software developmental process. While each of the models may vary slightly, it is generally agreed that software development consists of several qualitatively different activities. These distinct activities are requirement analysis, functional specification, architectural design, implementation, and evolution [Ref.15]. Inherent in each of these activities is the aspect of quality assurance. The flaw in the traditional “waterfall” software development approach is the late identification of any requirement errors. Prototyping as described in chapter 3 solves this flaw through an iterative process where the designer and user jointly develop the initial requirements and specification for the critical parts of the system. The requirements then serve as the basis to develop a prototype system that ultimately can be demonstrated to the customer. This chapter describes the software developmental process in terms of the software developmental activities for the prototype of the Fuel Automated Subsystem.

B. REQUIREMENTS ANALYSIS

The main purpose of requirement analysis is to identify a customer’s needs in sufficient detail to plan the construction of a software system to meeting those needs [Ref. 15]. Typically, a model of the systems environment is built that captures the problem the system must solve in order to refine and formalize requirements. The model assists in identifying the goals and constraints of the customer organization, which motivate the goals and constraints of the proposed software system [Ref. 15]. The specific results of requirements analysis are the following [Ref. 15]:

(1) a simplified model of the system's environment.

(2) a description of the goals of the system and its functionality.

(3) identification of performance, implementation and resource constraints.

In order to achieve these results, the initial problem must be formalized, unstated requirements discovered, and inconsistencies resolved. The general process of requirements analysis and its results are illustrated below:

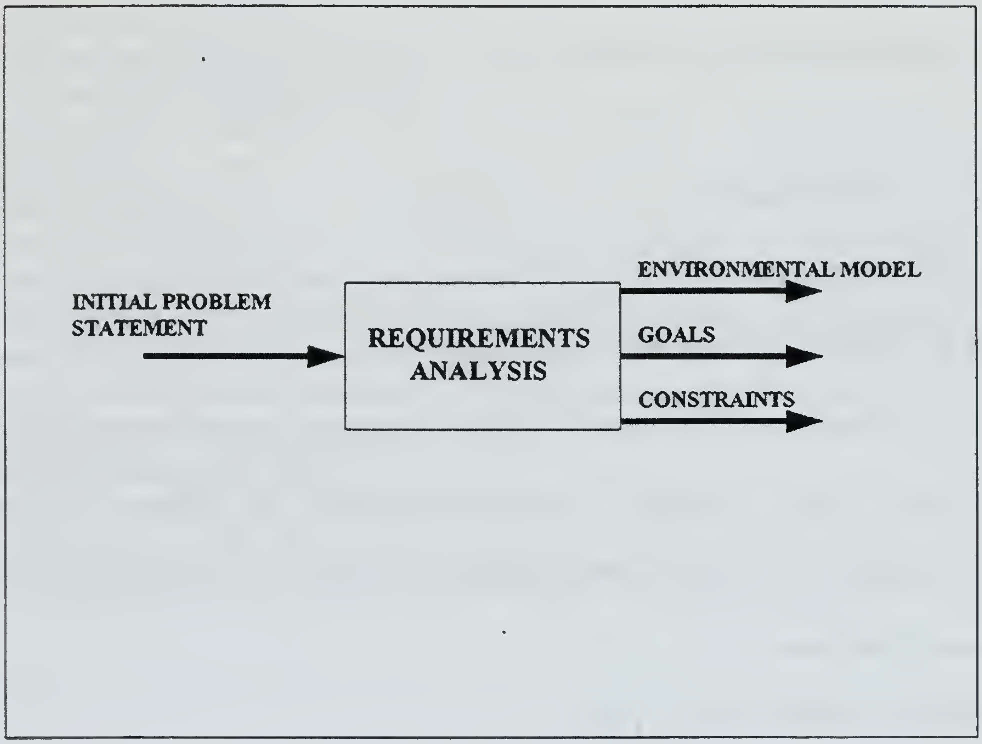


Figure 13. Derivation of Requirements [Ref. 15]

In the case of the Fuel Automated Subsystem, the developmental plan for the parent ICS3 system immediately limits the systems scope. Phase I of the ICS3 developmental plan specifically calls for the development of initial operational capability at the retail level. This requirement of establishing operational capability at the retail level serves as the initial informal problem statement for the Fuel Automated Subsystem. Since retail petroleum operations have not been automated and the informal problem

statement is vague, the major task is to establish a precise, testable, and feasible set of requirements. Accordingly, to determine the specific requirements for the Fuel Automated Subsystem a full analysis of the problem domain, retail petroleum operations, is required.

To fully analyze retail petroleum operations, a top down approach was applied. Through a review of applicable Army petroleum field manuals, technical manuals, and regulations, retail petroleum operations can be generally classified into three areas. These general areas of retail petroleum operations are as follows:

- *Personnel*: This area covers the personnel assigned to conduct retail petroleum operations. Examples of specific personnel functions include determining personnel status and manpower utilization.
- *Equipment*: This area covers the physical equipment required to perform petroleum operations. Examples of specific equipment functions include the following: maintaining accountability of equipment on-hand, ordering shortages, tracking maintenance and service schedules, and tracking licensing status.
- *Operational*: This area covers the core fuel processes of receiving, storing, issuing, as well as conducting quality surveillance.

A review of the general classification areas and functionality listed above against the designated modules of ICS3 reveals that the Personnel and Maintenance Modules of ICS3 fully address the areas of personnel and equipment. Currently, the STAMIS of SIDPERS and SAMS capture the personnel and maintenance functionality and will be incorporated into their respective ICS3 modules. Through interfaces within ICS3, the

retail petroleum user will be able to obtain required personnel and equipment information. However, the operational area functionality is not supported in any current STAMIS or ICS3 module and hence becomes the focal point for the development of the Fuel Automated Subsystem contained within the Supply Support Module.

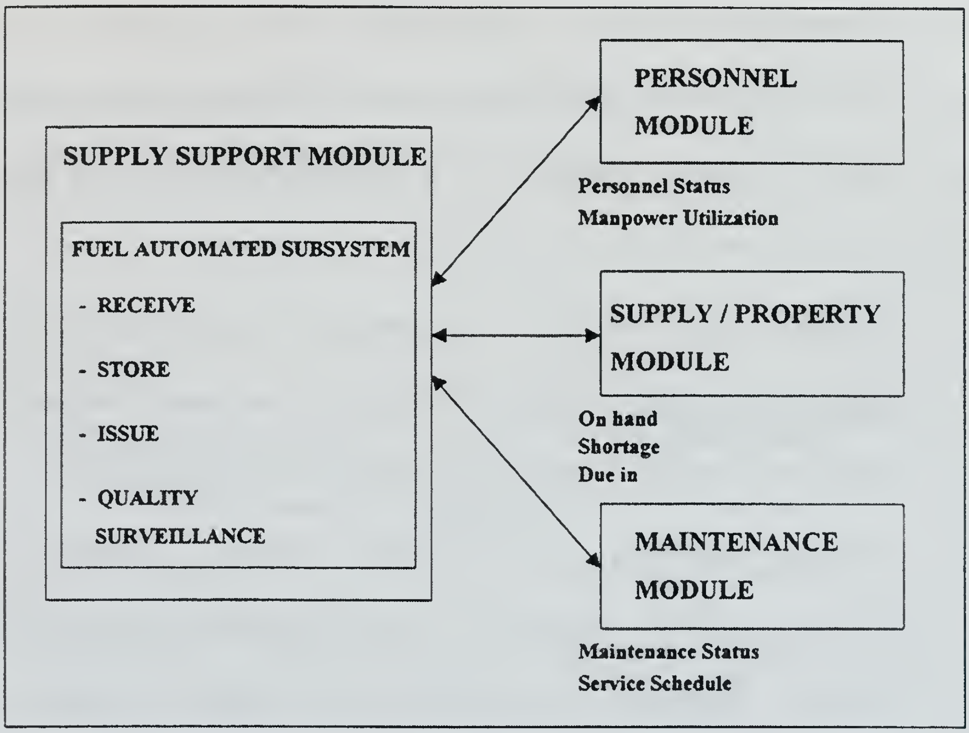


Figure 14. ICS3 Modules and Retail Petroleum Processes

A further in-depth analysis of the core petroleum processes of receiving, storing, issuing, and quality surveillance yields a set of further supporting functions and activities. A summary of the respective supporting retail petroleum activities and functions are listed below:

RECEIVE FUNCTIONS	STORE FUNCTIONS	ISSUE FUNCTIONS	QUALITY SURVEILLANCE FUNCTIONS
Schedule delivery	Determine ullage	Schedule issues	Fuel testing
Inspect receiving equipment	Account for inventory	Inspect equipment	Meter and lab instrumentation calibration

Inspect transporter	Sample fuel	Load vehicle, railcar, or aircraft	Schedule sampling and testing
Gage & sample fuel	Fuel circulation & consolidation	Account for issues	Relay lab results to customers
Unload vehicle, ship or tank car	Inspect facilities / storage containers	Process requisitions	Inspect fuel transporter
Receive from assault hoseline or pipeline	Conduct filter separator maintenance	Determine ullage	Test filter separator
Conduct quality assurance	Dispose of contaminated / interface fuel	Conduct quality assurance	Provide fuel disposition
Account for received fuel / inventory	Conduct quality assurance	Conduct financial accounting	
Conduct financial accounting	Conduct financial accounting		

Figure 15. Retail Petroleum Functions

A goal within the requirements analysis process is to determine an automation boundary; which functions will be solved by the initial version of the proposed system, which ones will be included in later versions of the system, and which ones will remain outside the proposed system. In the case of the functions of receive, store, issue and quality surveillance listed above, most are physical in nature and require a soldier to perform them. These physical tasks do not lend themselves well to automation as part of the initial version of the Fuel Automated Subsystem. However, inherent in all of the functions is the requirement for petroleum management that becomes the ultimate focus for the initial developmental version of the Fuel Automated Subsystem.

1. System Environment

The environmental model serves as the initial basis for communication and agreement between the customer and the developer. The environmental model

specifically defines the concepts and relationships needed for describing the world in which the proposed system will operate. These concepts consist of identifying the individual objects, object types, attributes of objects, and the relationships between objects. [Ref. 15]

Petroleum management involves maintaining accurate accounts and executing appropriate quality control procedures for all receipts, issues and stocks on hand. The current manual fuel management procedures described in Chapter 2, E and illustrated in Figure 8 can be used to describe an informal view of the system to be developed/automated as the Fuel Automated Subsystem. A formal model of a system can also be constructed by specifying a finite number of explicit definitions for types, attributes, relationships, and laws to help develop precise descriptions of complex problems. The immediate benefit of the formal environment definition is that it standardizes the vocabulary of the system and allows for computer aided checking and analysis. The formal environment model for the Fuel Automated Subsystem expressed in Spec, a formal specification language, is located in Appendix B. Spec is a formal specification language based on logic that is detailed in Software Engineering with Abstractions by Berzins and Luqi [Ref. 15].

2. System Goals

The requirements for the Fuel Automated Subsystem are further formalized by determining the goals of the system and the functionality it must perform. With the current petroleum management procedures and environmental model as a base line, a goal hierarchy was developed to guide the design and implementation of the prototype

for the Fuel Automated Subsystem. The high-level goals were derived from the initial problem statement and are as follows:

G1: The purpose of the system is to assist the petroleum specialist in maintaining accurate accountability of the receipt, storage and issue of petroleum products.

G1.1: The system must allow the petroleum specialist to manually input receipt and issue documentation (DD Form 1348-1 from supply sources).

G1.2: The system must allow the petroleum specialist to view the current physical quantity of each petroleum product being stored.

G1.3: The system must maintain a daily database of all input petroleum receipts and issues.

G1.4: The system must maintain a monthly database of the total daily petroleum receipts and issues.

G1.5: The system must allow the petroleum specialist to cancel an incorrectly input receipt or issue.

G2: The system must provide a means for the accountable officer to insure proper petroleum product accountability.

G2.1: The system must allow the accountable officer to view the current physical quantity of each petroleum product being stored.

G2.2: The system must provide a monthly accountability report.

Each of the top-level goals expresses the overall general objectives of the Fuel Automated Subsystem. Supporting second-level goals specify in greater detail how the

system is to accomplish the respective top-level goal. The complete hierarchical goal structure is located in Appendix C, and is structured with high level goals that have been refined into a set of supporting lower level goals. These refined sub-goals express a level of abstraction close to the proposed system.

3. System Constraints

Implementation, performance and resource constraints are typically examined as part of requirement analysis [Ref. 15]. The purpose of developing the Fuel Automated Subsystem prototype is to illustrate its viability as part of ICS3. As a result, the actual constraints on system development are very limited. However, the following general constraints by type were applied in the development of the Fuel Automated Subsystem to reflect the current Army operational environment:

- *Implementation:*

(1) The current petroleum documentation requirements and parameters defined in Army Regulation 710-2 can not be modified during implementation of the system.

(2) The system must be modular in design to capture the current documentation features. Daily and monthly receipt/issue databases must be incorporated as an audit trail as well as a means for future functionality such as petroleum trend analysis/forecasting.

(3) The user interface to the system must incorporate a format and structure that is easy to learn, efficient, and easy to remember.

(4) The use of CAPS as the prototyping tool for the development of the Fuel Automated Subsystem imposes certain developmental constraints.

Specifically, CAPS imposes an implementation constraint that the prototype system be developed in Ada, the programming language generated by CAPS.

- *Performance:*

(1) The responses of the Fuel Automated Subsystem must be fast enough as not to hinder fuel receipt and issue operations.

(2) The system must maintain 100% accuracy of both the physical and "book" quantity of fuel stored.

- *Resource:*

(1) The budget for final vendor implementation of the Fuel Automated Subsystem will be limited. The system must be developed for full use with currently fielded petroleum and personnel authorizations. No new equipment will be developed or personnel authorized to support this system.

(2) The system prototype must be fully developed by 1999 to comply with the ICS3 phased developmental plan.

4. Model Summary

A context summary provides a general overview of the proposed system. Specifically, the context summary lists the proposed system and shows the user classes and the types of objects the system controls [Ref. 15]. The context summary for the Fuel Automated Subsystem using the Spec language is as follows:

Fuel_Automated_Subsystem

Used_by : petroleum_specialist, accountable_officer

Controls : individual_fuel_receipts, individual_fuel_issues, consolidated_daily_log_of_receipts/issues, monthly_log_of_all_daily_receipts/issues, monthly_accountability_calculations

Figure 16. Fuel Automated Subsystem Context Summary

Additionally, the context summary can be illustrated in graphical form. The graphical context summary serves as a valuable tool for expressing the proposed system to individuals unfamiliar with the system's scope. The graphical context summary for the Fuel Automated Subsystem is as follows:

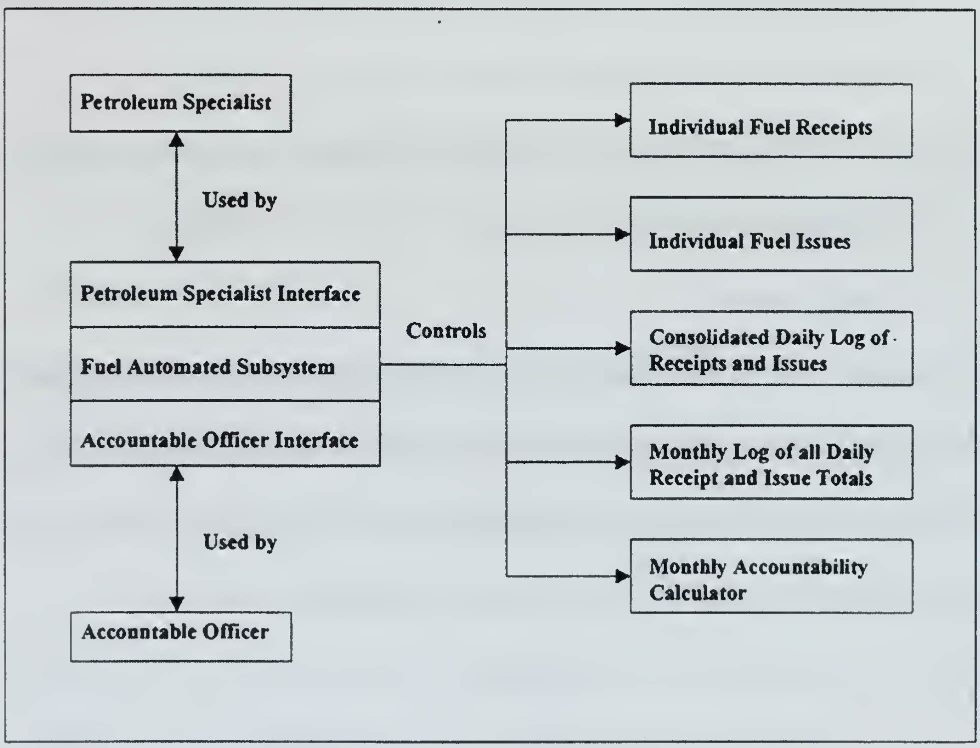


Figure 17. Fuel Automation Subsystem Context Diagram

C. PROTOTYPE DESIGN

In the prototyping cycle, a prototype system is designed from the result of requirements analysis. Inherent in the design of a prototype is the determination of both the functional specification and architectural design of the system. The functional specification describes the external interfaces along with the concepts needed to use the proposed system, while the architectural design describes both the internal and external interfaces along with the concepts needed to build the proposed system [Ref. 15]. The main goal of architectural design activity is to progressively decompose the proposed system into a smaller set of independent modules [Ref. 15]. Large systems must be decomposed into modules that enable a team of developers to work on different parts of the system concurrently. Through concurrent development, each team can conduct independent analysis and design of individual modules while developing an accurate estimation of costs and schedules. The modules must be decomposed through many levels of abstraction until the sub-modules either match existing reusable software components or are simple enough to be directly implemented. The important goals of the decomposition are feasibility and ease of implementation. The specific results of architectural design are the following [Ref. 15]:

(1) Modules of the decomposed system should be understandable in isolation.

(2) All interactions between modules should be explicitly defined.

(3) Each module should be small and simple.

1. Modules

In accordance with the activity of architectural design, this section analyzes the particular requirements of the individual modules of the Fuel Automated Subsystem. A

brief description of the attributes of each module identified in the Army petroleum management/accountability process is included. The listed modules only represent the critical modules required for development of the prototype system. Additional modules will be required in later system development to enhance the overall systems capability.

- *Bulk Receipt Graphical User Interface:*

  The bulk receipt graphical user interface allows a petroleum specialist to manually input parameters extracted from DD Form 1348-1 during the receipt of bulk petroleum. The petroleum specialist enters the type of fuel received, quantity in gallons, and the document number of the receipt as required by Army Regulation 710-2.

- *Other Receipt Graphical User Interface:*

  The other receipt graphical user interface captures all other possible petroleum receipt scenarios (i.e. vehicle/aircraft defueling, etc.). The graphical user interface allows a petroleum specialist to manually enter the type of fuel received, quantity in gallons, an identification number from the source, and the source unit as required by Army Regulation 710-2.

- *Bulk Issue Graphical User Interface:*

  The bulk issue graphical user interface allows a petroleum specialist to manually input parameters extracted from DA Form 2765-1 during the issue of bulk petroleum. The petroleum specialist enters the type of fuel issued, quantity in gallons, the document number of the issue, the receiving unit, and the name/rank of the receiver as required by Army Regulation 710-2.

- *Other Issue Graphical UserInterface:*

The other issue graphical user interface allows a petroleum specialist to manually input parameters required by Army Regulation 710-2 for all other fuel issues other than bulk issues. These other issues include fuel issues made directly into, or specifically identifiable to, a consuming end item. An example of this type of issue is to a vehicle or an M2 burner unit. The petroleum specialist enters the type of fuel issued, quantity in gallons, the receiving vehicle bumper number/equipment name, the receiving unit, and the name/rank of the receiver.

- *Daily Receipt and Issue Consolidator:*

The daily receipt and issue consolidator serves as a the processor of all user input receipts and issues for diesel, mogas and jet fuel types. The receipt and issue consolidator stores all receipts and issues in a database as a historical audit trail. The receipt and issue consolidator also maintains a daily totalizer of all receipts and issues per fuel type.

- *Diesel Storage Tank:*

The diesel storage tank module emulates the physical quantity of diesel fuel on-hand and available for issue. Physical storage quantity is measured in gallons.

- *Jet Storage Tank:*

The jet storage tank module emulates the physical quantity of jet fuel on-hand and available for issue. Physical storage quantity is measured in gallons.

- *Mogas Storage Tank:*

The mogas storage tank module emulates the physical quantity of unleaded fuel on-hand and available for issue. Physical storage quantity is measured in gallons.

- *Monthly Receipt and Issue Consolidator:*

The monthly receipt and issue consolidator serves as a processor of the daily total receipts and issues for diesel, mogas, and jet fuel types. The monthly receipt and issue consolidator stores the daily total receipts and issues per fuel type in a database as a historical audit trail. The receipt and issue consolidator also maintains a monthly totalizer of all receipts and issues per fuel type.

- *Monthly Accountability Report Generator:*

The monthly accountability report generator utilizes the physical quantity of fuel stored on the first day of the month, the monthly receipts and issues, and the physical quantity of fuel stored on the last day of the month to calculate the monthly gain/loss and allowable gain/loss. If actual loss exceeds the allowable loss, the accountability report generator signals the accountable officer.

- *Accountable Officer Graphical User Interface:*

The accountable officer user interface allows the accountable officer to monitor the current quantity of fuels on hand and the monthly accountability report.

2. Basic Model

Based upon the identified modules, a basic architectural model of the Fuel Automated Subsystem was constructed. The systems architecture for daily fuel accountability/management functionality is illustrated as follows:

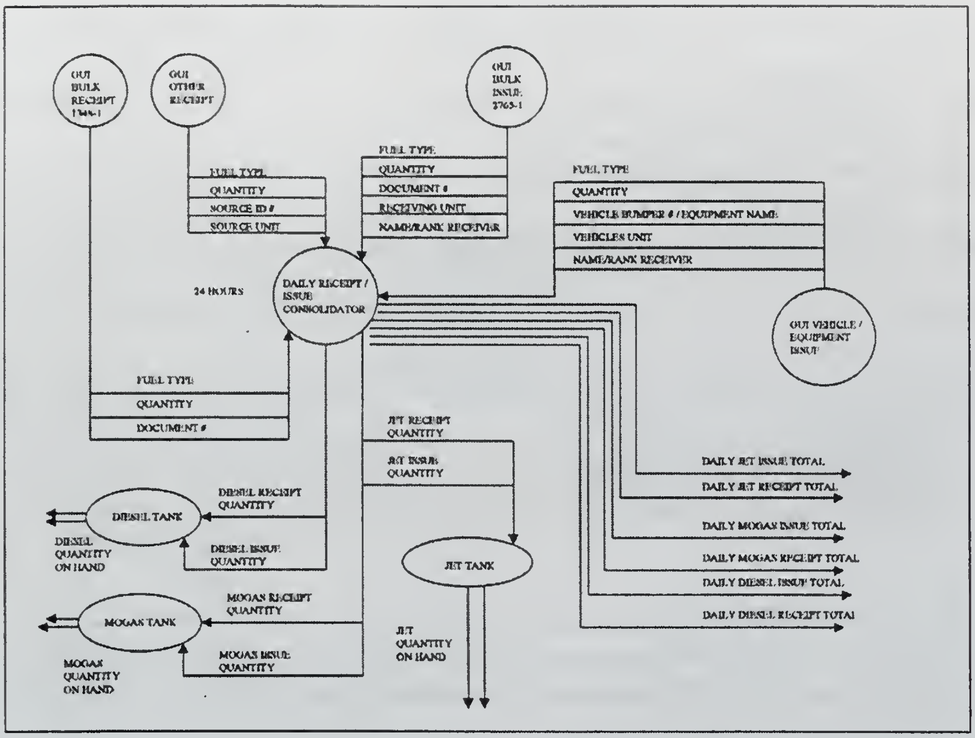


Figure 18. Basic Model Design for Daily Fuel Accountability

The systems architecture for monthly fuel accountability/management is illustrated as follows:

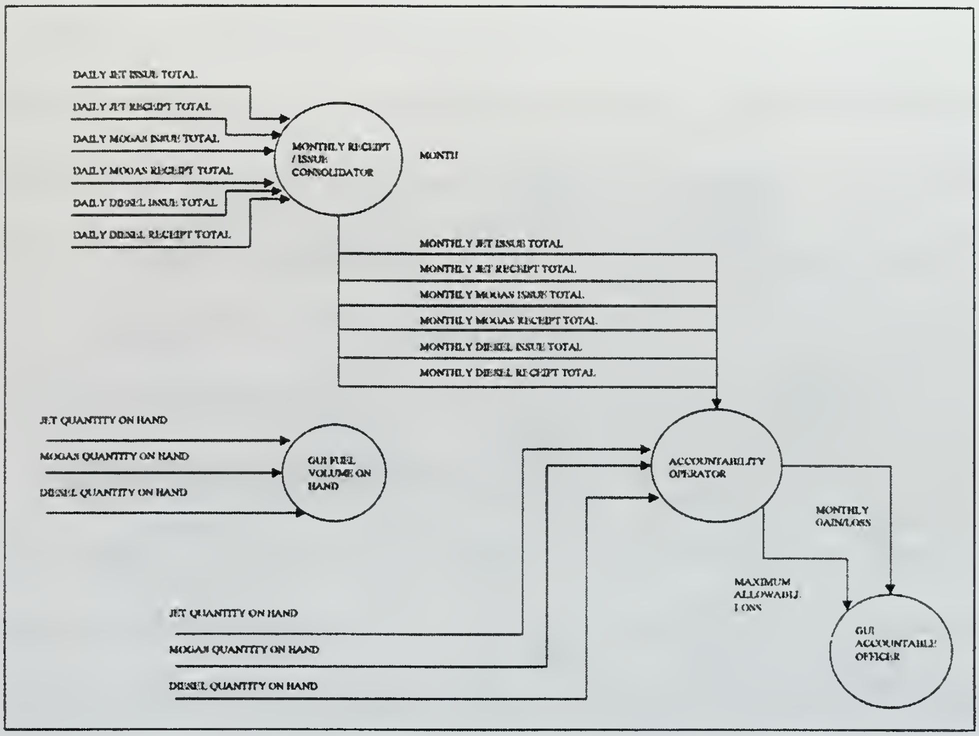


Figure 19. Basic Model Design for Monthly Fuel Accountability

From these initial models, the Fuel Automated Subsystem software architecture was constructed using CAPS.

D. CAPS PROTOTYPE

The basic model developed in the previous section identified the Fuel Automated Subsystem's high level functional operators, their interaction, and corresponding data flow. Using the basic model as a guide, the system architecture for the Fuel Automated Subsystem prototype was developed using CAPS.

1. Graphical Editor

Within the CAPS graphical editor, an initial high-level system architecture for the Fuel Automated Subsystem was developed. In the graphical editor, round bubbles represent proposed software modules, rectangular boxes indicate external systems interacting with the proposed software, and connecting streams represent interactions

between internal and/or external modules. The high-level architecture was further decomposed to reach atomic level operators that demonstrated specific functionality which could easily be implemented. The complete high-level system architecture and subsequent operator decompositions for the Fuel Automated Subsystem are included in Appendix D. The following high-level components were used in the CAPS graphical editor to model the Fuel Automated Subsystem. Each component's role and properties are briefly described below.

- **gui_bulk_receipt**

  This is a non-time critical operator that represents the GUI for bulk receipt parameter input. The GUI provides a means of manual input of fuel type, quantity and document number. Accordingly, the operator produces output streams of bulk_rcpt_fuel_type, bulk_rcpt_qty, and bulk_rcpt_doc_number. The bulk_rcpt_fuel_type stream is an integer type, corresponding to a radio button selection. The bulk_rcpt_qty stream is an integer type since all fuel receipt quantities are measured in whole gallons. The bulk_rcpt_doc_number stream is a string type to allow for alphanumeric input.

- **gui_other_receipt**

  This is a non-time critical operator that represents the GUI for all other fuel receipt input. The GUI provides a means of manual input of fuel type, quantity, a source identification number, and the sources unit. Accordingly, the operator produces output streams of oth_rcpt_fuel_type, oth_rcpt_qty, oth_rcpt_source, and oth_rcpt_source_unit. The oth_rcpt_fuel_type stream is an integer type, corresponding to a radio button selection. The oth_rcpt_qty stream is also an integer type. The oth_rcpt_source and oth_rcpt_source_unit streams are string types to allow for alphanumeric input.

- **gui_bulk_issue**

  This is a non-time critical operator that represents the GUI for bulk issue parameter input. The GUI provides a means of manual input of fuel type, quantity, document number, receiving unit, and the name of the receiver. Accordingly, the operator produces output streams of bulk_iss_fuel_type, bulk_iss_qty, bulk_iss_doc_num, bulk_rcv_unit, and bulk_rcv_name. The bulk_iss_fuel_type stream is an integer type, corresponding to a radio button selection. The bulk_iss_qty is also an integer type since all issues are

measured in whole gallons. The bulk_iss_doc_number, bulk_rcv_unit, and bulk_rcv_name are string types to allow for alphanumeric input.

- **gui_other_issue**

  This is a non-time critical operator that represents the GUI for all other fuel issue parameter input. The GUI provides a means of manual input of fuel type, quantity, identification number/nomenclature of the receiving piece of equipment, owning unit, and the name of the receiver. Accordingly, the operator produces output streams of eq_iss_fuel_type, eq_iss_qty, eq_iss_id, eq_iss_unit, and eq_iss_name. The eq_iss_fuel_type stream is an integer type, corresponding to a radio button selection. The eq_iss_qty is also an integer type. The eq_iss_id, eq_iss_unit and eq_iss_name streams are string types that allow for alphanumeric input.

- **daily_consolidator**

  This is a composite operator that processes and stores all daily fuel receipt and issue transactions. The daily_consolidator receives all of the streams generated from the four receipt and issue GUIs as input. Within the daily consolidator, there are two similar processing paths that ultimately converge at the daily_reporter operator. The first path is for receipts, and begins with the bulk_rcpt_processor, and oth_rcpt_processor. The other path is for issues, and begins with the bulk_iss_processor, and oth_iss_processor. Each of these processors receives the fuel type and quantity streams from their corresponding GUI as input. With the fuel type and quantity data present, the processor produces an enable signal that triggers either the receipt or issue processor depending on the type of transaction. The receipt or issue processor determines the fuel type and produces a stream to the respective storage tank to update the storage volume and a stream to the respective fuel type totalizer. The receipt or issue totalizers receive the new quantity and add the value to a corresponding state stream that holds the daily total. The state stream serves as input to the daily_reporter, a periodic operator that executes every hour, with a maximum execution time and finish within time of 750 msec. The normal period for the daily_reporter would be 24 hours, but was proportionately reduced to one hour to demonstrate the prototype. The daily_reporter produces a stream capturing each fuel types' daily total receipts and issues, and also reinitializes all of the totalizers. Also within the daily_consolidator is a relational database composed of four tables that capture all of the streams produced from the four GUIs as a historical audit trail.

- **monthly_consolidator**

  This is a composite operator that tallies and stores all daily receipts and issues. The monthly_consolidator receives all of the daily fuel type receipt and issue

totals produced by the daily_reporter. Within the monthly_consolidator are six monthly totalizers for the receipt or issue per fuel type. Each monthly totalizer receives a new daily total quantity and adds the value to a corresponding state stream that holds the monthly total. The state streams serve as input to the monthly_reporter, a periodic operator that executes every 30 hours, with a maximum execution time and finish within time of 750 msec. The normal period for the monthly_reporter would be 720 hours (approximately one month), but was proportionately reduced to 30 hours to demonstrate the prototype. The monthy_reporter produces a stream capturing each fuel types' monthly total receipts and issues, and also reinitializes all of the monthly totalizers. Also within the monthly_consolidator is a relational database composed of two tables that capture all of the daily receipt and issue totals per fuel type as a historical audit trail.

- **accountability_op**

  This is a composite operator that calculates monthly fuel accountability. Within the accountability_op are three accountability calculators, one per fuel type. Each accountability calculator receives the respective total monthly receipts, total monthly issues, the current fuel quantity available in storage, and the monthly opening inventory as inputs. From these streams, the accountability calculator determines the monthly gain/loss, allowable gain/loss, and compares the two values. Based upon the comparison, the accountability calculator produces a boolean tolerance output stream.

- **gui_acc_officer**

  This is a non-time critical operator that represents the GUI for displaying monthly accountability results to the accountable officer. The GUI receives the boolean streams tolerance_jet, tolerance_df, and tolerance_mg representing whether the monthly accountability tolerance per fuel type has been met.

- **gui_fuel_on_hand**

  This is a non-time critical operator that represents the GUI for displaying the current amounts of diesel, mogas, and jet fuel on-hand and available for issue. The operator receives the diesel_qty_available, mogas_qty_available, and jet_qty_available streams that represent the gallon quantity of each fuel in storage as input. A gaging operator within each respective fuel tank external operator produces these streams.

- **diesel_tank**

  This is an external composite operator representing the stored quantity of diesel fuel. The composite operator has input streams, diesel_rcpt_qty and

diesel_iss_qty, that represent integer quantities of diesel fuel which have been received or issued. Within this composite operator are the external atomic operators of diesel_addition and diesel_subtraction. The diesel_addition operator adds the diesel_rcpt_qty stream value to the diesel_volume state stream value to simulate an increase in diesel fuel storage volume due to a receipt. The diesel_subtraction operator subtracts the diesel_iss_qty stream value from the diesel_volume state stream value to simulate a decrease in diesel fuel storage volume. Both the addition and subtraction operators produce an output integer state stream, df_qty_on_hand. The diesel gage operator receives the df_qty_on_hand stream and simulates the physical gaging of the diesel storage tanks to determine the quantity available for issue. Accordingly, the diesel_gage operator outputs the integer stream diesel_qty_available.

- **mogas_tank**

  This is an external composite operator representing the stored quantity of mogas. The composite operator has input streams, mogas_rcpt_qty and mogas_iss_qty, that represent integer quantities of mogas which have been received or issued. Within this composite operator are the external atomic operators of mogas_addition and mogas_subtraction. The mogas_addition operator adds the mogas_rcpt_qty stream value to the mogas_volume state stream value to simulate an increase in mogas fuel storage volume due to a receipt. The mogas_subtraction operator subtracts the mogas_iss_qty stream value from the mogas_volume state stream value to simulate a decrease in mogas storage volume. Both the addition and subtraction operators produce an output integer state stream, mg_qty_on_hand. The mogas gage operator receives the mg_qty_on_hand stream and simulates the physical gaging of the mogas storage tanks to determine the quantity available for issue. Accordingly, the mogas_gage operator outputs the integer stream mogas_qty_available.

- **jet_tank**

  This is an external composite operator representing the stored quantity of jet fuel. The composite operator has input streams, jet_rcpt_qty and jet_iss_qty, that represent integer quantities of jet fuel which have been received or issued. Within this composite operator are the external atomic operators of jet_addition and jet_subtraction. The jet_addition operator adds the jet_rcpt_qty stream value to the jet_volume state stream value to simulate an increase in jet fuel storage volume due to a receipt. The jet_subtraction operator subtracts the jet_iss_qty stream value from the jet_volume state stream value to simulate a decrease in jet fuel storage volume. Both the addition and subtraction operators produce an output integer state stream, jet_qty_on_hand. The jet gage operator receives the jet_qty_on_hand stream and simulates the physical gaging of the jet storage tanks to determine the

quantity available for issue. Accordingly, the jet_gage operator outputs the integer stream jet_qty_available.

Further in-depth atomic operator descriptions can be found in the prototype description language code included in Appendix E.

2. Prototyping Description Language

From the completed graphical representation of the Fuel Automated Subsystem, CAPS automatically generated a prototype description language (PSDL) prototype. The PSDL prototype is composed of the set of operators identified in the graphical editor's system graphs. Every PSDL operator is unambiguously defined through a specification and implementation section [Ref. 11]. The specification section defines the operators interface and provides an informal description of the operator's behavior [Ref. 11]. The implementation section contains either an architectural description that defines the decomposition of composite operators or code interface descriptions for atomic operators [Ref. 11]. The Fuel Automated Subsystem's complete PSDL code is included in Appendix E.

3. Translating and Scheduling

The PSDL program serves as the basis for generating compilable and executable prototype code. The CAPS translator successfully transformed the Fuel Automated Subsystem PSDL program in Appendix E into Ada code that implemented all supervisory aspects of the prototype. Specifically, the CAPS translator generated the CAPS support packages and the main prototype procedure that implemented the data streams, execution guards, output guards, operation triggers and timers [Ref. 16]. The support package and main procedure ultimately became part of the CAPS generated supervisor module fuel_subsystem.a.

Next, the CAPS scheduler was successfully evoked to create the remaining portions, the static and dynamic schedules, of the fuel_subsystem.a module. The CAPS scheduler specifically used the timing information contained within the PSDL program to determine schedule feasibility, and to create the respective code to implement the schedule. Using the earliest deadline scheduling algorithm, the CAPS scheduler determined a feasible schedule for the Fuel Automated Subsystem. CAPS implemented time critical operators in the static schedule, while non-time critical operators were implemented in the dynamic schedule. Both the static and dynamic schedules created by the CAPS scheduler completed the fuel_subsystem.a supervisor module. The fuel_subsystem.a supervisor module for the Fuel Automated Subsystem is included in Appendix F.

4. Implementation

The CAPS translation and scheduling process constructed Ada template files/packages for all user-defined types and atomic operators. From the template files/packages, Ada implemetation files were written for all user-defined types and atomic operators. The only user-defined data type used in the Fuel Automated Subsystem prototype was text_string. The type text_string was declared to allow the system user to input alphanumeric parameters such as unit name/numbers, document numbers, etc., from the graphical user interfaces. The text_string type was declared as a subtype of type string with a length of 100 characters.

Also from the template files, all atomic operators were implemented in Ada to simulate the expected behavior of the operator. With the user-defined text string declared, all atomic operators implemented, and the fuel_subsystem.a supervisor module,

the CAPS compiler (Sun/Ada) succeeded in generating executable code for the Fuel Automated Subsystem with no errors or warnings. To test the system's simple functionality, the periods of the daily_reporter and monthly_reporter were reduced, and all atomic operators were enhanced to print operator specific messages to the prototype console window. The complete Ada code for the fuel_subsystem.a supervisor module, the atomic operators without the test message stream output, and the text_string declaration is in Appendix F.

To further augment the Fuel Automated Subsystem prototype, proposed GUIs for bulk receipts, other receipts, bulk issues, other issues, fuel on hand, and monthly accountability were constructed using TAE+, a graphical user interface generator. While simple in construct, the GUIs present a tangible image of the system to the user. The GUIs incorporate the required CAPS input parameters for fuel receipts and issues. Also, the GUIs display the available quantity of fuel, and display whether monthly accountability tolerances are being met in accordance with the CAPS output parameters. The proposed GUIs for the Fuel Automated Subsystem prototype are included in Appendix G. After customer review, the proposed GUIs may be further modified and/or integrated into the CAPS code.

V. CONCLUSIONS AND RECOMMENDATIONS

A. SIGNIFICANCE OF THE PROTOTYPE

The Fuel Automated Subsystem represents the initial step in the automation of the combat service support discipline of “fuel” as mandated in the February 1996, TRADOC Pamphlet 525-5, Force XXI Operations. As a result of a complete analysis of current Army petroleum operations, the task of automating petroleum accountability/management was identified as the core process in any fuel automation system. Within the constraints of the ICS3 developmental process, the scope of automating petroleum accountability/management was further restricted to retail operations. With the target domain of retail petroleum accountability/management identified, the requirements analysis process produced a goal hierarchy and environmental model for the proposed system.

From the developed goal hierarchy and environmental model, a prototype of the Fuel Automated Subsystem was successfully developed using CAPS to illustrate the systems’ viability. The CAPS approach for implementing the Fuel Automation Subsystem prototype provided an integrated set of tools that permitted the detailed specification, design, and implementation of the prototype system in a single developmental environment. The CAPS developmental environment also provided the ability to simultaneously make prototype design changes to the PSDL specification and the Ada prototype source code. Through the use of the graphical user interface generator, TAE+, proposed GUIs were developed to present a tangible image of the system to the user.

The initial Fuel Automated Subsystem prototype developed in this thesis can now be reviewed and verified by the customer, the Combat Developments Branch at CASCOM. Based upon the feedback from the Combat Development Branch, the prototype can be modified accordingly to reflect any new customer requirements. This iterative prototyping process can continue until the customer is ultimately satisfied with the system.

B. RECOMMENDED FUTURE WORK

The work completed on the CAPS Fuel Automated Subsystem leaves open the possibility of future system improvements and enhancements. Specifically, the following prioritized list addresses potential areas of future work and research:

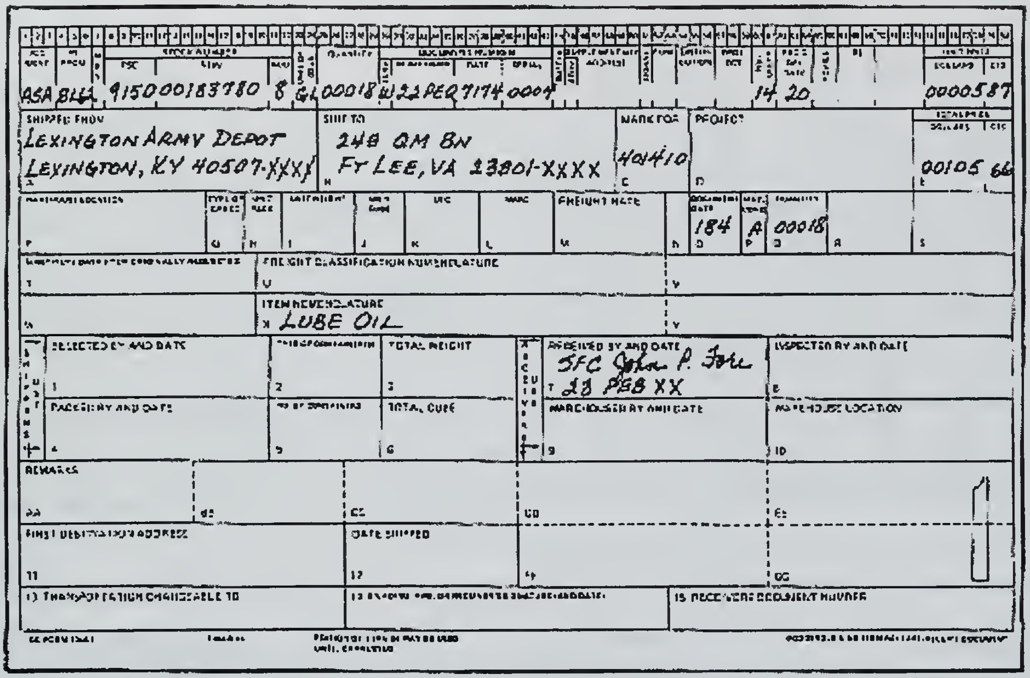
- The TAE+ graphical user interfaces contained in Appendix G need to be validated and integrated into the CAPS prototype Ada code in order to fully evaluate the Fuel Automated Subsystem's performance and suitability.
- Exception guards within the various operators should be added to enhance the prototype's fault tolerance.
- The daily and monthly relational database operators in the prototype have not been fully implemented. Currently, the operators take the input stream parameters and assign them to variables in order to simulate storage in a database. A commercially available database should be integrated into the prototype to more closely simulate the desired system functionality.
- The current Fuel Automated Subsystem prototype does not address hardware issues such as the system's target hardware platform. Once a target platform is identified, the CPU ratio within CAPS should be modified to reflect the

target platform and the prototype's performance can be re-tested and reevaluated.

- The current Fuel Automated Subsystem prototype has not incorporated any interaction with other ICS3 modules. Once other ICS3 module interfaces are clearly defined, the Fuel Automated Subsystem must be integrated. For example, the Fuel Automated Subsystem must be inter-linked with the ICS3 financial module to insure that the cost of fuel issues are deducted from unit budgets.
- While the Fuel Automated Subsystem addresses the critical issues revolving around petroleum management/accountability, the prototype leaves room for other additional applications and extensions. For example, additional functionality such as the ability to perform fuel consumption trend analysis could be included.

APPENDIX A

The following fuel management documents and reports are currently in use:



ASA BLL 9150001837780 8 GL00018 W22PEQ7174 0004 14 20 0000587

SHIPPED FROM
LEXINGTON ARMY DEPOT
LEXINGTON, KY 40507-XXXX

SHIP TO
248 QM BN
FT LEE, VA 23801-XXXX

MARK FOR
W01410

00105 66

184 A 00018

FREIGHT CLASSIFICATION NOMENCLATURE

ITEM NOMENCLATURE
LUBE OIL

SELECTED BY AND DATE

TOTAL WEIGHT

RECEIVED BY AND DATE
SFC John P. Fore
28 FEB XX

INSPECTED BY AND DATE

PACKED BY AND DATE

TOTAL CUBE

WAREHOUSED BY AND DATE

WAREHOUSE LOCATION

REMARKS

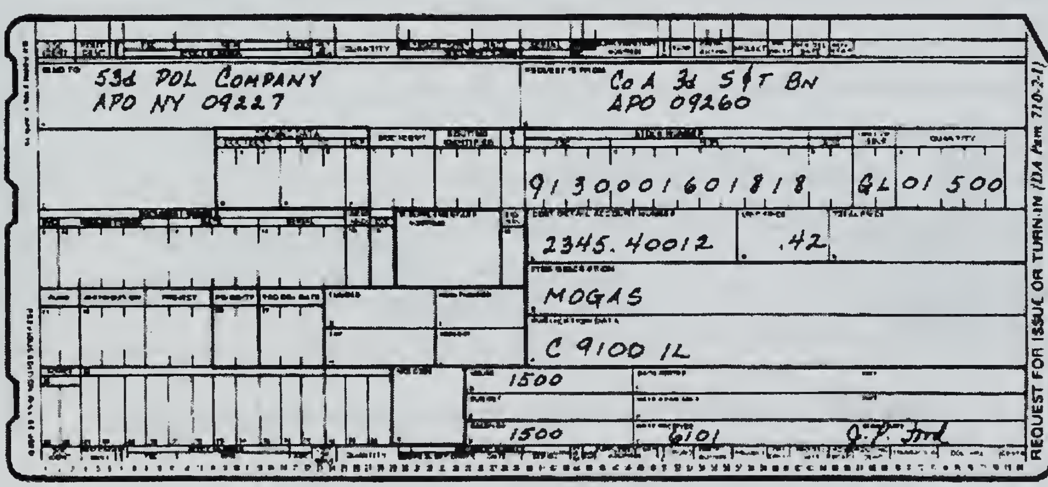
FIRST DESTINATION ADDRESS

DATE SHIPPED

13 TRANSPORTATION CHARGEABLE TO

15 RECEIVER'S DOCUMENT NUMBER

Figure 1. DD Form 1348-1: Single Line Item Release / Receipt Document



53d POL COMPANY
APO NY 09227

Co A 3d S&T BN
APO 09260

9130001601818 GL 01 500

2345.40012 .42

MOGAS

C 9100 IL

1500

1500 6101 J.P. Ford

REQUEST FOR ISSUE OR TURN-IN (DA Pam 710-2-1)

Figure 2. DA Form 2765-1: Request for Issue or Turn-in

DAILY ISSUES OF PETROLEUM PRODUCTS
For use of this form, see AR 703-1; the proponent agency is DCSLOG.

PAGE NO. 1 | NO. OF PAGES 1

VEHICLE USA REGISTRATION NUMBER (a)	ISSUES GAL: MOGAS (b)	ISSUES GAL: DIESEL (c)	ISSUES GAL: (d)	RECEIPTS GAL: MOGAS (e)	RECEIPTS GAL: DIESEL (f)	RECEIPTS GAL: (g)	ORGANIZATION AND ADDRESSES (Indicate Service: A, Army, AF, Air Force, N, Navy, M, Marine Corps) (h)	SIGNATURE GRADE (i)
91390012				500				P. Swift
183201	21						1st Bde Ft Bright, SC – A	J. James, PFC
91390047					500			P. Swift
1A2521		42					Marine Corps Air Station Cherry Point, NC – M	[illegible], Cpl
91390112					500			P. Swift
182800	10						3d Arty Ft Bright, SC – A	E. Shay, SP4
TOTAL RECEIPTS				500	1000			
TOTAL ISSUES	31	42						

POST, CAMP OR STATION	DATE	SIGNATURE OF ATTENDANT
Ft Bright, SC	15 May 1986	Robert T. Williams

DA FORM APR 85 3643 — EDITION OF 1 OCT 70 IS OBSOLETE.

Figure 3. DA Form 3643: Daily Issues of Petroleum Products

MONTHLY ABSTRACT OF ISSUES OF PETROLEUM PRODUCTS AND OPERATING SUPPLIES				POST, CAMP OR STATION Ft Bright, SC				MONTH May 1986			VOUCHER NO. 0006		
DATE	Insert type, grade and unit of issue for each product issued (e.g., engine oil, OE 30, quart)												
	ISSUES (GALS)						RECEIPTS (GALS)						
	MO a	JP b	DF c	OTHER d	OTHER e	OTHER f	MO g	JP h	DF i	OTHER j	OTHER k	OTHER l	OTHER m
1	404	0	0				7500	0	0				
2	0	0	0				0	5000	0				
3	3031	15	1489				0	0	0				
4	2201	0	0				0	0	0				
5	4133	725	4740				0	0	7500				
6	3682	120	4740				5000	5000	7500				
7	7297	200	6839				10000	15000	0				
8	1239	4283	1137				7500	0	15000				
9	0	0	0				0	0	0				
10	0	0	0				0	0	0				
11	1977	0	0				0	0	0				
12	0	0	0				0	[illegible]	0				
13	404	1728	4780				5000	0	7500				
14	744	2743	911				500	5000	0				
15	224	10336	655				0	30000	15000				
16	0	0	0				0	0	0				
17	15	0	466				0	0	0				
18	2538	3456	1298				0	0	0				
19	0	0	0				0	0	0				
20	783	239	443				5000	0	0				
21	1123	8334	783				0	7500	0				
22	227	343	3478				0	0	7500				
23	157	0	2247				12500	5000	5000				
24	0	0	0				0	0	0				
25	0	0	0				0	0	0				
26	8977	15778	7523				0	0	0				
27	665	2123	1227				0	0	30000				
28	121	1157	780				0	12500	0				
29	363	987	433				500	0	0				
30	0	735	87				0	0	0				
31	112	483	141				0	5000	0				
TOTAL	41377	53807	44177				53500	90000	95000				
TOTAL GAL¹	41377	53807	44177				53500	90000	95000				
SIGNATURE OF ACCOUNTABLE PROPERTY OFFICER Robert Boyd PBO WAFCAA						GRADE 1LT		POSTED TO STOCK RECORD ACCOUNT BY Juan E. Rodriguez			DATE 1 June 1986		

¹To convert oil, lubricating, to gallons, divide total quarts by 4. To convert gear lube to gallons, divide total pounds by 7½.

DA FORM 3644, APR 85 — REPLACES DA FORM 3644, 1 OCT 70, WHICH WILL BE USED. ☆ GPO: 1970 380-288 1018

Figure 4. DA Form 3644: Monthly Abstract of Issues of petroleum Products and Operating Supplies

MONTHLY BULK PETROLEUM ACCOUNTING SUMMARY

POST, CAMP OR STATION | PROPERTY ACCOUNT NUMBER | PERIOD OF REPORT FROM ______ TO ______

PRODUCTS						
Stock Number						
Nomenclature	MOGAS	DIESEL	JP-4			
OPENING INVENTORY	145,000	110,000	170,000			
RECEIPTS	130,000	125,000	220,000			
ISSUES	85,000	105,000	185,000			
CLOSING BOOK BALANCE	190,000	130,000	205,000			
PHYSICAL CLOSING INVENTORY	188,500	129,500	202,500			
MONTHLY GAIN/LOSS	1,500	500	2,500			
MAXIMUM ALLOWABLE LOSS — GASOLINE & JP-4	2,750		3,900			
MAXIMUM ALLOWABLE LOSS — OTHER FUELS		1,175				

REMARKS

NAME & GRADE OF ACCOUNTABLE OFFICER	SIGNATURE	DATE
THOMAS G. THOMAS, 1LT	Thomas G. Thomas	1 AUGUST 19XX
NAME & GRADE OF APPROVING OFFICER	SIGNATURE	DATE

DA FORM 4702-R APR 85 EDITION OF 1 JUN 78 IS OBSOLETE

Figure 5. DA Form 4702-R: Monthly Bulk Petroleum Accounting Summary

APPENDIX B

Software Engineering with Abstractions, Appendix C by Luqi and Berzins provides definitions of predefined Spec concepts. Some of these predefined concepts are inherited and imported to develop an environmental model of the Fuel Automated Subsystem. The Fuel Automated Subsystem environmental model is as follows:

```
DEFINITION fuel automation_subsystem_environment
        INHERIT system  -- defines software_system, proposed, controls
        INHERIT user  -- defines User_class, uses
        INHERIT business  -- defines product
        INHERIT person -- defines person
        INHERIT cause -- defines Needed_for
        IMPORT subtype FROM type

        CONCEPT fuel_automated_subsystem : software_system
                WHERE proposed (fuel_automation_subsytem),
                          -- Will construct a fuel automated subsystem
                        Controls (fuel_automation_subsystem, petroleum_management)
                          -- System will help Army personnel manage petroleum

        CONCEPT petroleum_specialist : User_class
                WHERE ALL (ps : petroleum_specialist ::
                                  uses (ps, fuel_automated_subsystem))
                        Subtype (petroleum_specialist, person)
                          -- Petroleum specialists use the fuel automated subsystem

                        ALL (f : fuel ::
                                SOME (ps : petroleum_specialist :: manages (ps, f)))
                          -- For every fuel, there is a petroleum specialist that manages it

                          -- Petroleum specialists are the only source of bulk and retail fuel
                          -- issues and receipt documentation

                        ALL (fr : DD_1348_1 ::
                                SOME (ps : petroleum_specialist :: receive_bulk (ps, fr)))
                          -- For every bulk fuel receipt document, there is a petroleum
                          -- specialist that received the bulk fuel
```

```
ALL (fi : DA_2765_1 ::
        SOME (ps : petroleum_specialist :: issue_bulk (ps, fi)))
  -- For every bulk fuel issue document, there is a petroleum
  -- specialist that issued the bulk fuel

ALL (rr : retail_receipt_doc ::
        SOME (ps : petroleum_specialist :: receive_retail (ps, rr)))
  -- For every retail receipt document, there is a petroleum
  -- specialist that received the retail fuel

ALL (ri : retail_issue_doc ::
        SOME (ps : petroleum_specialist :: issue_retail (ps, ri)))
  -- For every retail issue document, there is a petroleum specialist
  -- that issued the retail fuel

CONCEPT issue_bulk (uc : User_class, ad : accountability_document)
    VALUE (b : boolean)
     -- True if the user class issues bulk fuel in accordance with
     -- DA Pam 710-2-1

CONCEPT receive_bulk (uc : User_class, ad : accountability_document)
    VALUE (b : boolean)
      -- True if the user class receives bulk fuel in accordance with
      -- DA Pam 710-2-1

CONCEPT issue_retail (uc : User_class, ad : accountability_document)
    VALUE (b : boolean)
      -- True if the user class issues retail fuel in accordance with
      -- DA Pam 710-2-1

CONCEPT receive_retail (uc : User_class, ad : accountability_document)
    VALUE (b : boolean)
      -- True if the user class receives retail fuel in accordance with
      -- DA Pam 710-2-1

CONCEPT DD_1348_1 : type
    WHERE Subtype (DD_1348_1, accountability_document)
        ALL (rbf : to_receive_bulk_fuel :: Needed_for (DD_1348_1, rbf)
          -- A DD Form 1348-1 is needed to receive bulk fuel

CONCEPT to_receive_bulk_fuel : type
    WHERE Subtype (to_receive_bulk_fuel, activity)
        ALL (rbf : to_receive_bulk_fuel :: Quantity (rbf) > 500)
          -- To receive fuel greater than 500 gallons is a bulk receipt
```

```
CONCEPT DA_2765_1 : type
        WHERE Subtype (DA_2765_1, accountability_document)
                ALL (ibf : to_issue_bulk_fuel :: Needed_for (DA_2765_1,ibf)
                  -- A DA Form 2765-1 is needed to issue bulk fuel

CONCEPT to_issue_bulk_fuel : type
        WHERE Subtype (to_issue_bulk_fuel, activity)
                ALL (ibf : to_issue_bulk_fuel :: Quantity (ibf) > 500)
                  -- To issue fuel greater than 500 gallons is a bulk issue

 CONCEPT retail_receipt_doc : type
        WHERE Subtype (retail_receipt_doc, accountability_document)
                ALL (rrf : to_receive_retail_fuel ::
                        Needed_for (retail_receipt_doc, rrf)
                   -- Retail receipt documentation is needed to receive retail fuel

CONCEPT to_receive_retail_fuel : type
        WHERE Subtype (to_receive_retail_fuel, activity)
                ALL (rrf : to_receive_retail_fuel :: Quantity (rrf) ≤ 500)
                  -- To receive fuel less than or equal to 500 gallons is a retail
                  -- receipt

CONCEPT retail_issue_doc : type
        WHERE Subtype ( retail_issue-doc, accountability_document)
                ALL (irf : to_issue_retail_fuel ::
                         Needed_for (retail_issue_doc, irf)
                   -- Retail issue documentation is needed to issue retail fuel

CONCEPT to_issue_retail_fuel : type
        WHERE Subtype (to_issue_retail_fuel, activity)
                ALL (irf : to_issue_retail_fuel :: Quantity (irf) ≤ 500)
                   -- To issue fuel less than or equal to 500 gallons is a retail issue

CONCEPT DA_3643 : type
        WHERE Subtype (DA_3643, accountability_document)
                  Captures_all_daily (DA_3643, DD_1348_1, DA_2765_1,
                        retail_issue, retail_receipt )
                    -- Daily log and totalizer of all fuel receipts and issues

CONCEPT DA_3644 : type
        WHERE Subtype (DA_3644, accountability_document)
                  Captures_daily_totals (DA_3644, DA_3643)
                    -- Monthly log and totalizer of the total daily receipts and
                    -- issues from DA 3643
```

```
CONCEPT DA_4702 : type
        WHERE Subtype (DA_4702, accountability_document)
                  Calculates_accounting_from (DA_3644, opening_inventory,
                                                   physical_inventory)
                   -- Accountability calculator that uses the months opening fuel
                   -- inventory, monthly total receipts / issues from the DA 3644,
                   -- and closing physical inventory to determine if accountability
                   -- is within specified tolerance and to adjust the following
                   -- months opening inventory

CONCEPT accountable_officer : User_class
        WHERE ALL (ao : accountable_officer ::
                          uses(ao, fuel_automated_subsystem))
                Subtype (accountable_officer, person)
                Monitors (accountable_officer, fuel_automated_subsystem)
                 -- Accountable officers use and monitor the system
                Audits (accountable_officer, DA_4702)
                 -- Accountable officers audit the monthly DA Form 4702

CONCEPT monitors (uc : User_class, s : system)
        VALUE (b : boolean)
         --  True if the instance of the user class can be created

CONCEPT quantity (a : activity)
        VALUE (r : real)
        -- The amount of fuel issued or received

CONCEPT accountability_document : type
        -- Accounting documents used for supply transactions in accordance with
        -- DA Pam 710-2-1

CONCEPT fuel : type
        WHERE Subtype (fuel, product)
          --  There are three types of fuel products

CONCEPT diesel_fuel : type
        WHERE Subtype (diesel_fuel, fuel)

CONCEPT jet_fuel : type
        WHERE Subtype (jet_fuel, fuel)

CONCEPT motor_fuel : type
        WHERE Subtype (motor_fuel, fuel)

END
```

The following diagram illustrates the critical concepts and relationships defined in the environmental model above.

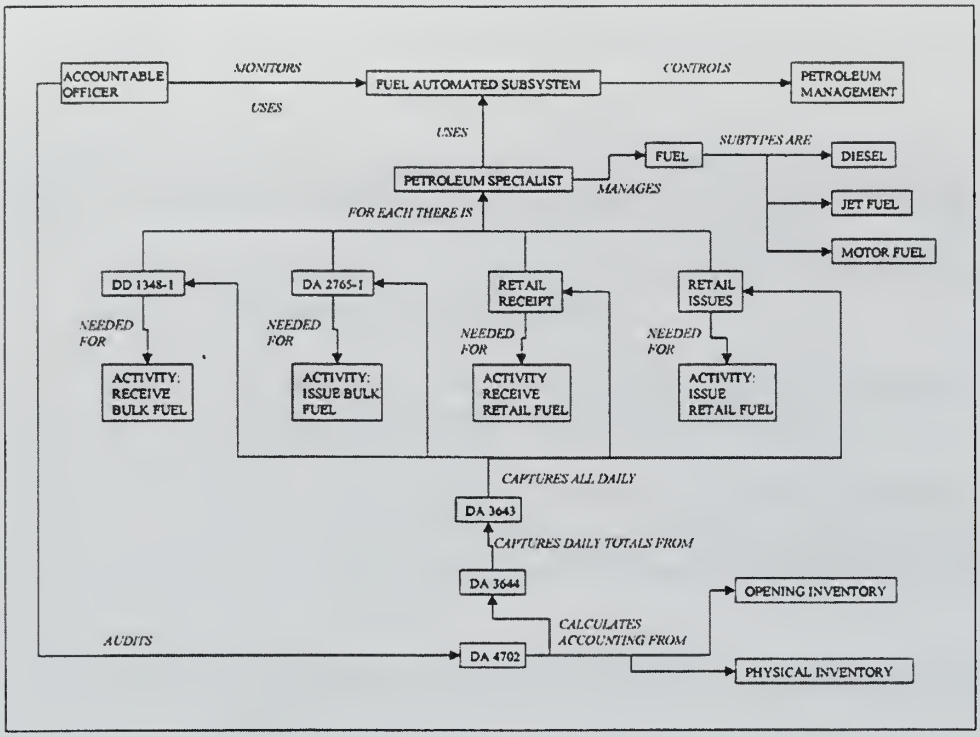


Figure 1. Relational Diagram of Environmental Model

APPENDIX C

The hierarchical goal structure for the Fuel Automated Subsystem is as follows:

G1: The purpose of the system is to assist the petroleum specialist in maintaining accurate accountability of the receipt, storage and issue of petroleum products.

G1.1: The system must allow the petroleum specialist to manually input receipt and issue documentation into the system.

G1.1.1: The system must allow for the input of bulk fuel receipt documentation (DD Form 1348-1 from supply source).

G1.1.1.1: The system must allow for input of the following parameters: name/rank of soldier making the receipt, type of fuel received, quantity in gallons, document number.

G1.1.2: The system must allow for the input of fuel receipts from other sources.

G1.1.2.1: The system must allow for input of the following parameters: name/rank of the soldier making the receipt, type of fuel received, quantity in gallons, source: vehicle/equipment identification number, source unit.

G1.1.3: The system must allow for the input of issue documentation for issues made into a transportation vehicle or storage tank that will subsequently be issued to a consuming end item of equipment (DA Form 2765-1).

G1.1.3.1: The system must allow for input of the following parameters: name/rank of the soldier making the issue, type of fuel issued, quantity in gallons, document number, unit of the receiver, name/rank of receiver.

G1.1.4: The system must allow for the input of issues made directly into or specifically identifiable to a consuming end item (i.e. vehicle).

G1.1.4.1: The system must allow for input of the following parameters: name/rank of the soldier making the issue, type of fuel issued, quantity in gallons, receiving vehicle bumper number, receiving vehicle's unit, name/rank of receiver.

G1.1.5: The system must allow for the input of issues made directly into an identifiable piece of equipment other than a vehicle.

G1.1.5.1: The system must allow for input of the following parameters: name/rank of the soldier making the issue, type of fuel issued, quantity in gallons, name of receiving equipment, equipment's unit, name/rank of receiver.

G1.2: The system must allow the petroleum specialist to view the current physical quantity of each petroleum product being stored.

G1.2.1: The system must receive as input, physical quantity reports in gallons from the storage tanks for each fuel type.

G1.3: The system must maintain a daily database of all input petroleum receipts and issues.

G1.3.1: The system must store all receipt and issue input fields in the database.

G1.3.2: The system must generate a daily receipt and issue total based on the individual receipts and issues for each fuel type being stored.

G1.4: The system must maintain a monthly database of the total daily petroleum receipts and issues.

G1.4.1: The system must store the total daily receipt and issue totals per fuel type in the database.

G1.4.2: The system must generate a monthly receipt and issue total based on the daily receipt and issue totals for each fuel type being stored.

G1.5: The system must allow the petroleum specialist to cancel an incorrectly input receipt or issue.

G2: The system must provide a means for the accountable officer to insure proper petroleum product accountability.

G2.1: The system must allow the accountable officer to view the current physical quantity of each petroleum product being stored.

G2.1.1: The system must receive as input, physical quantity reports in gallons from the storage tanks for each fuel type.

G2.2: The system must provide a monthly accountability report.

G2.2.1: The system must calculate the maximum monthly allowable gain/loss for each fuel type stored during the month. The monthly allowable gain/loss for jet fuel and mogas must be calculated as follows:

monthly allowable gain/loss = (opening inventory
+ total monthly receipts) x 0.01

and for diesel fuel as follows:

monthly allowable gain/loss = (opening inventory
+ total monthly receipts) x 0.005

G2.2.2: The system will calculate the actual monthly gain/loss for each fuel type stored during the month. The monthly gain/loss must be calculated for each respective fuel as follows:

monthly gain/loss = (opening inventory for the month
+ total monthly receipts
- total monthly issues)
- physical closing inventory

G2.2.3: The system must identify when the monthly allowable gain/loss is exceeded.

APPENDIX D

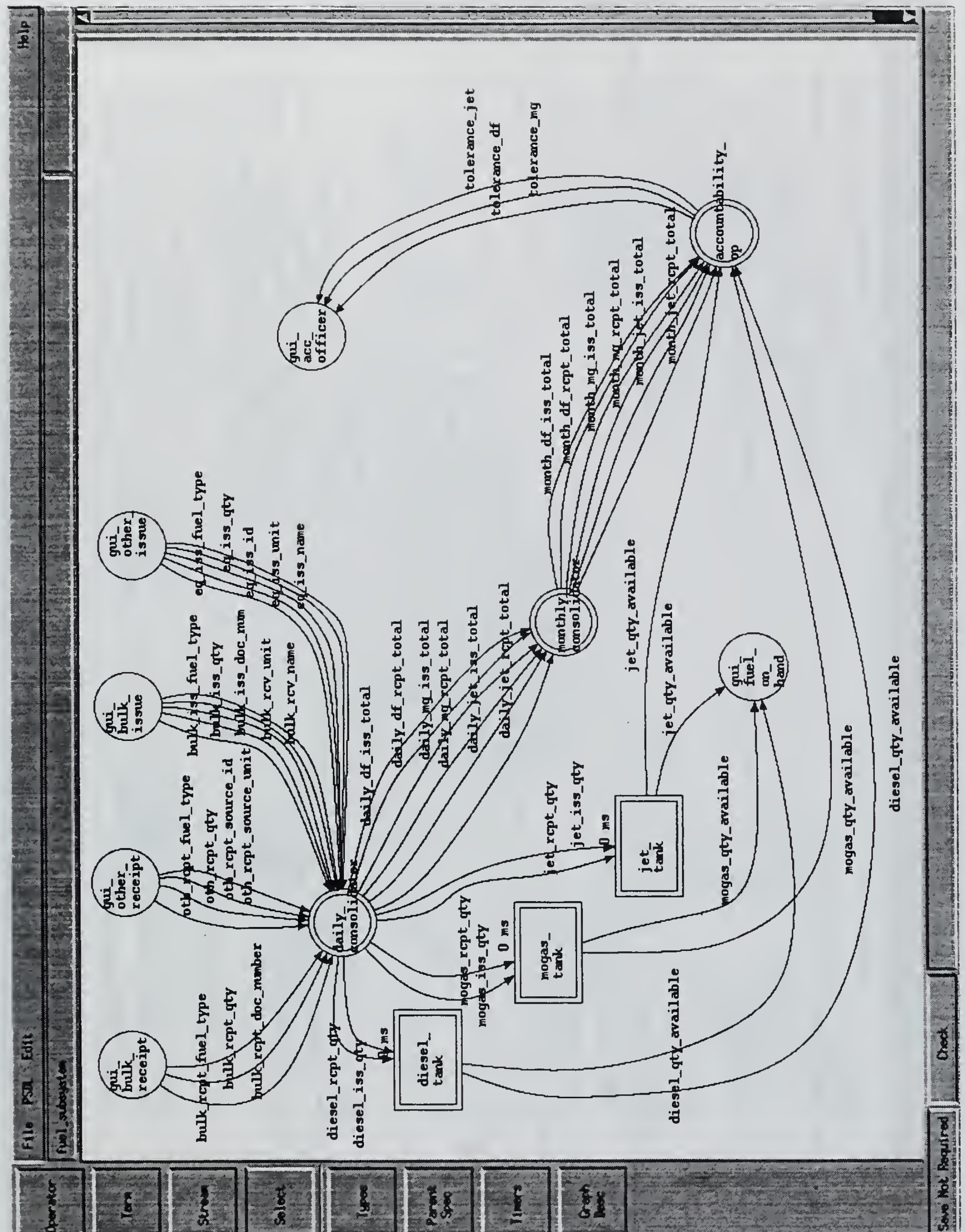


Figure 1. Top Level Graph of the Fuel Automated Subsystem

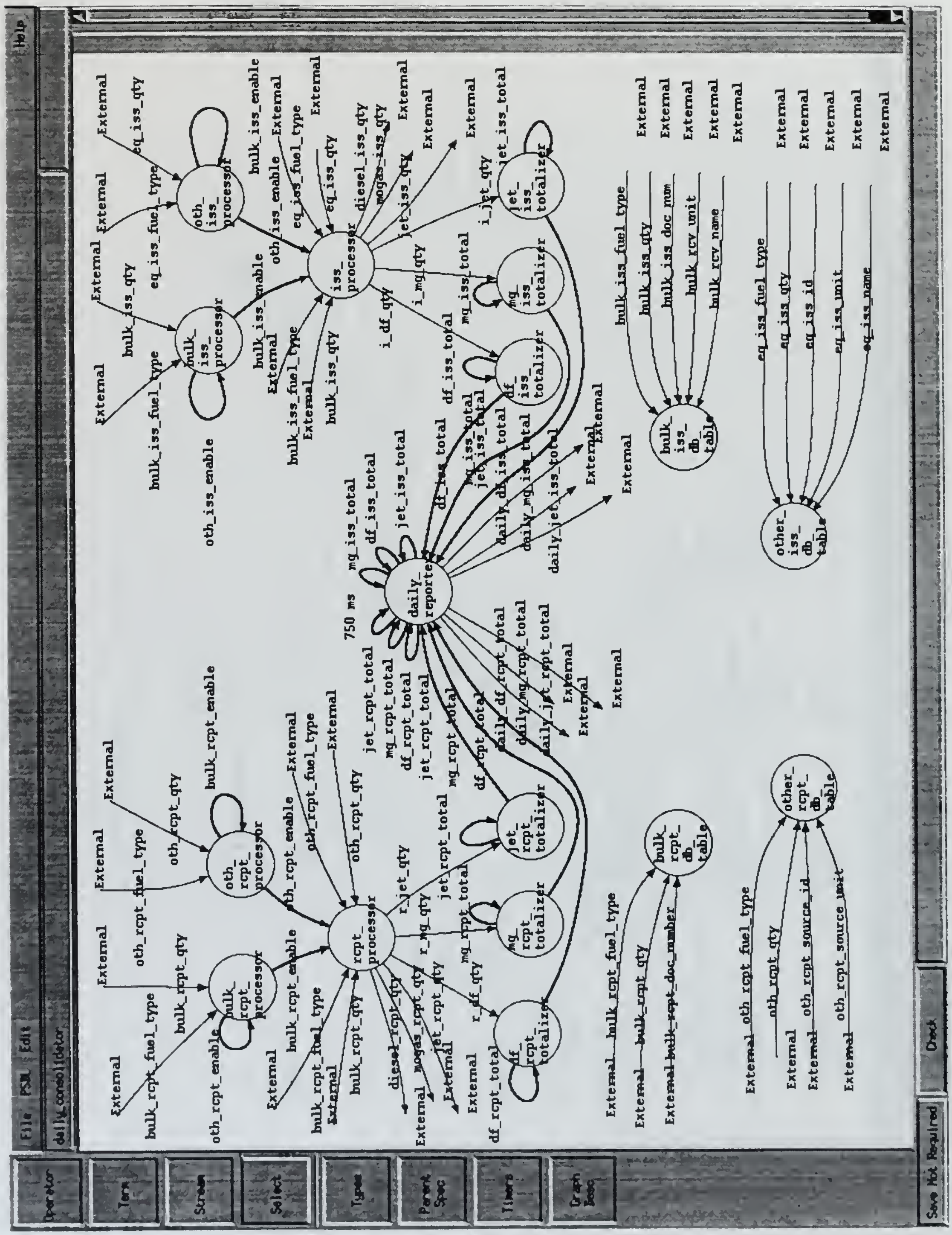


Figure 2. Decomposed Daily Consolidator

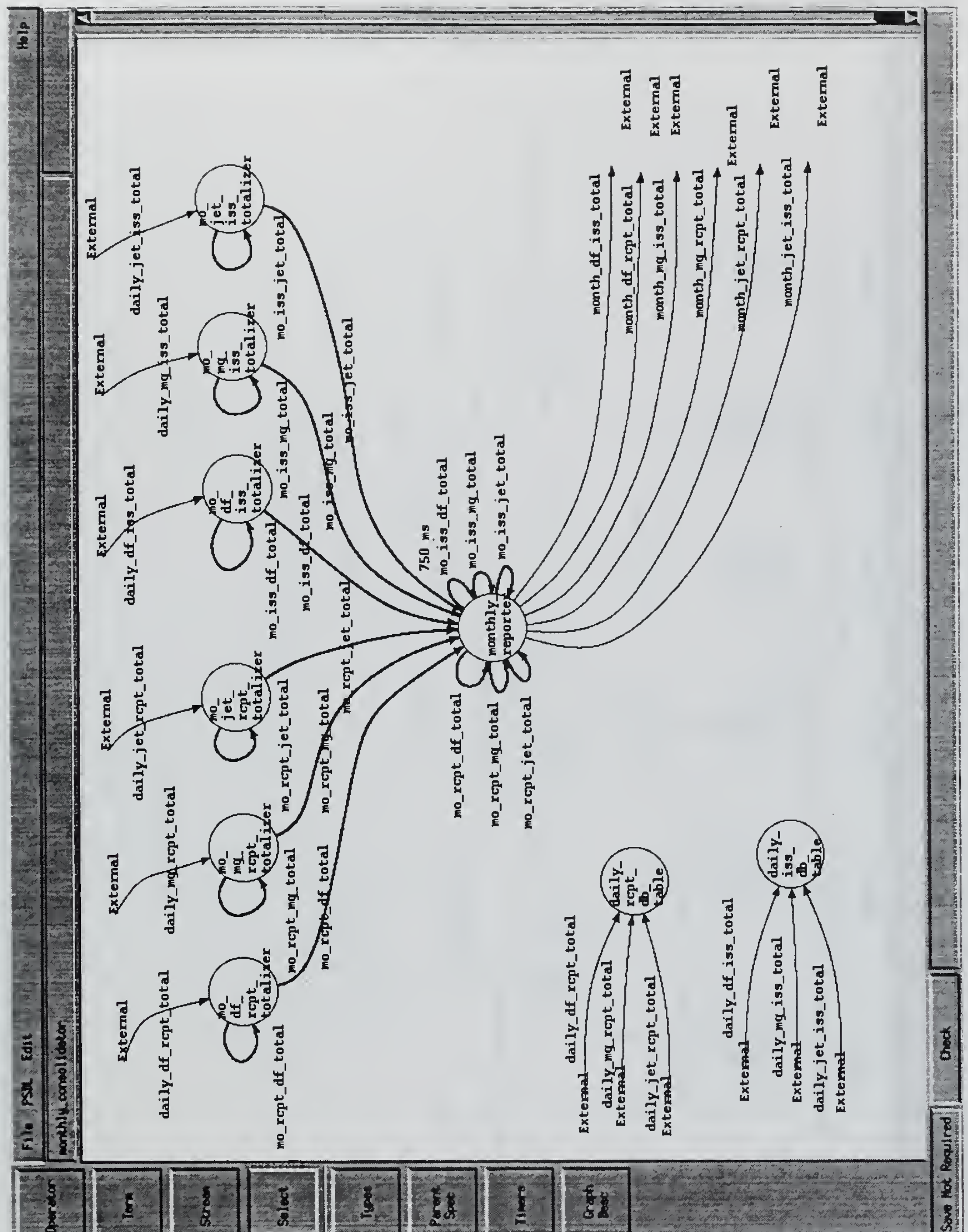


Figure 3. Decomposed Monthly Consolidator

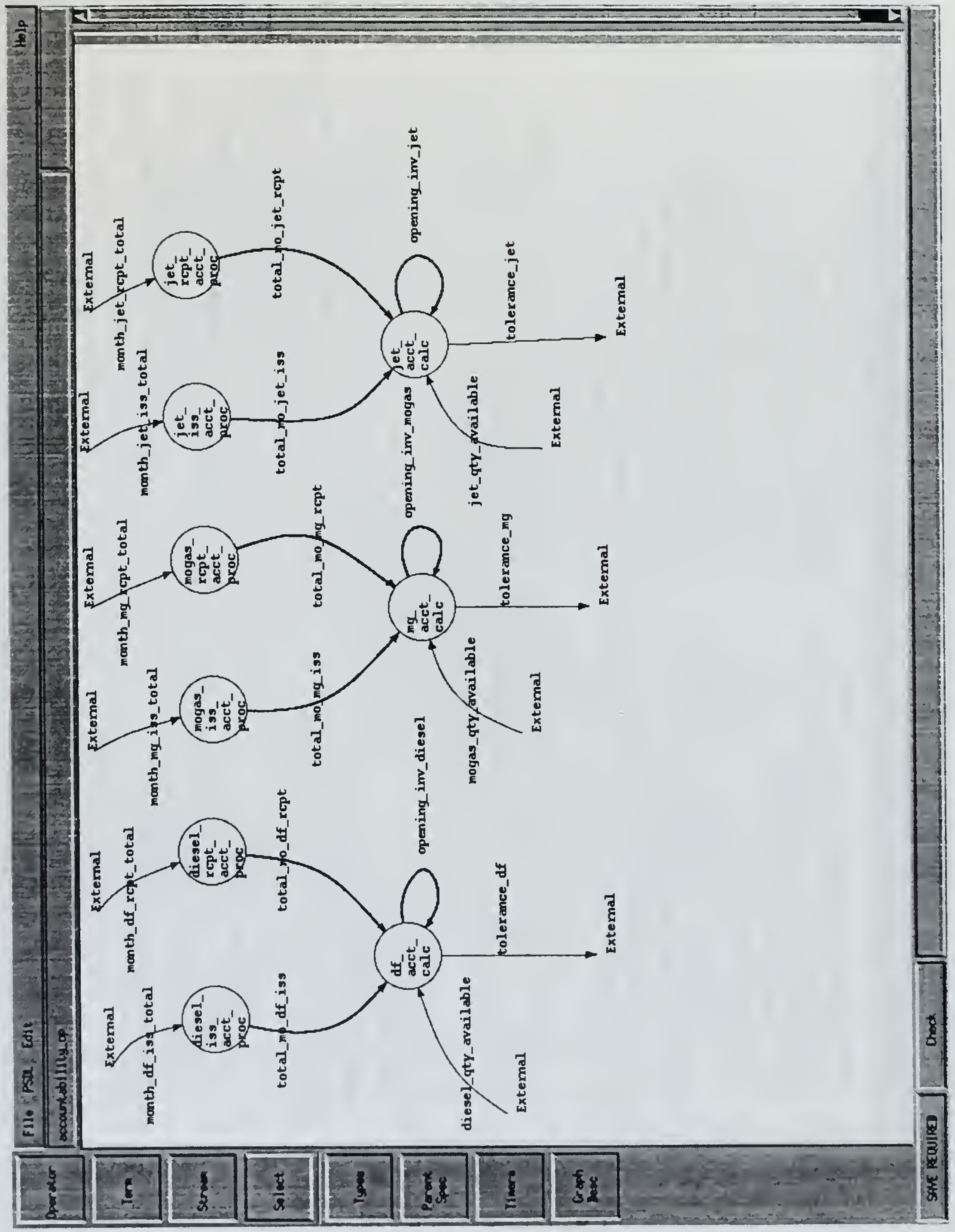


Figure 4. Decomposed Accountability Operator

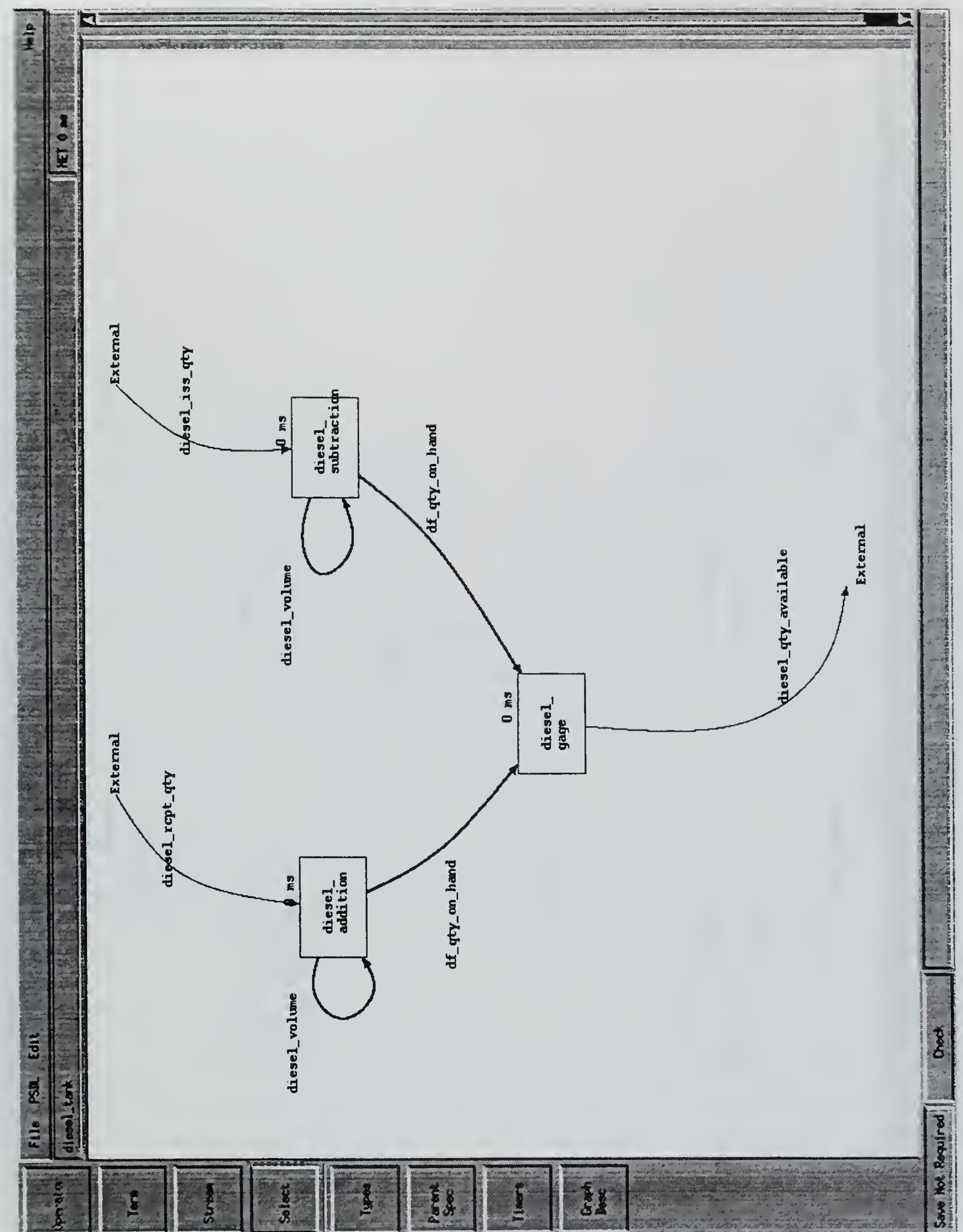


Figure 5. Decomposed Diesel Fuel Tank

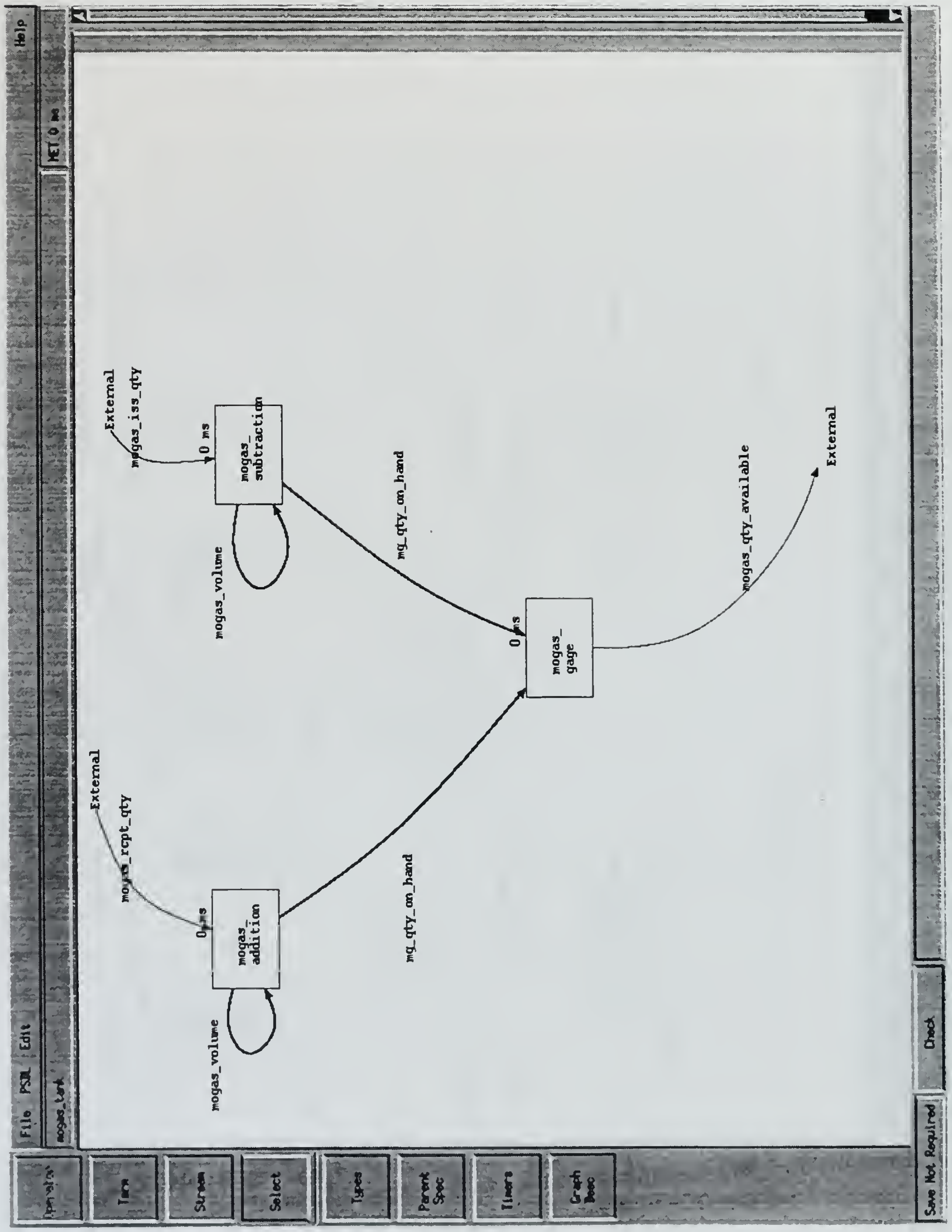


Figure 6. Decomposed Mogas Fuel Tank

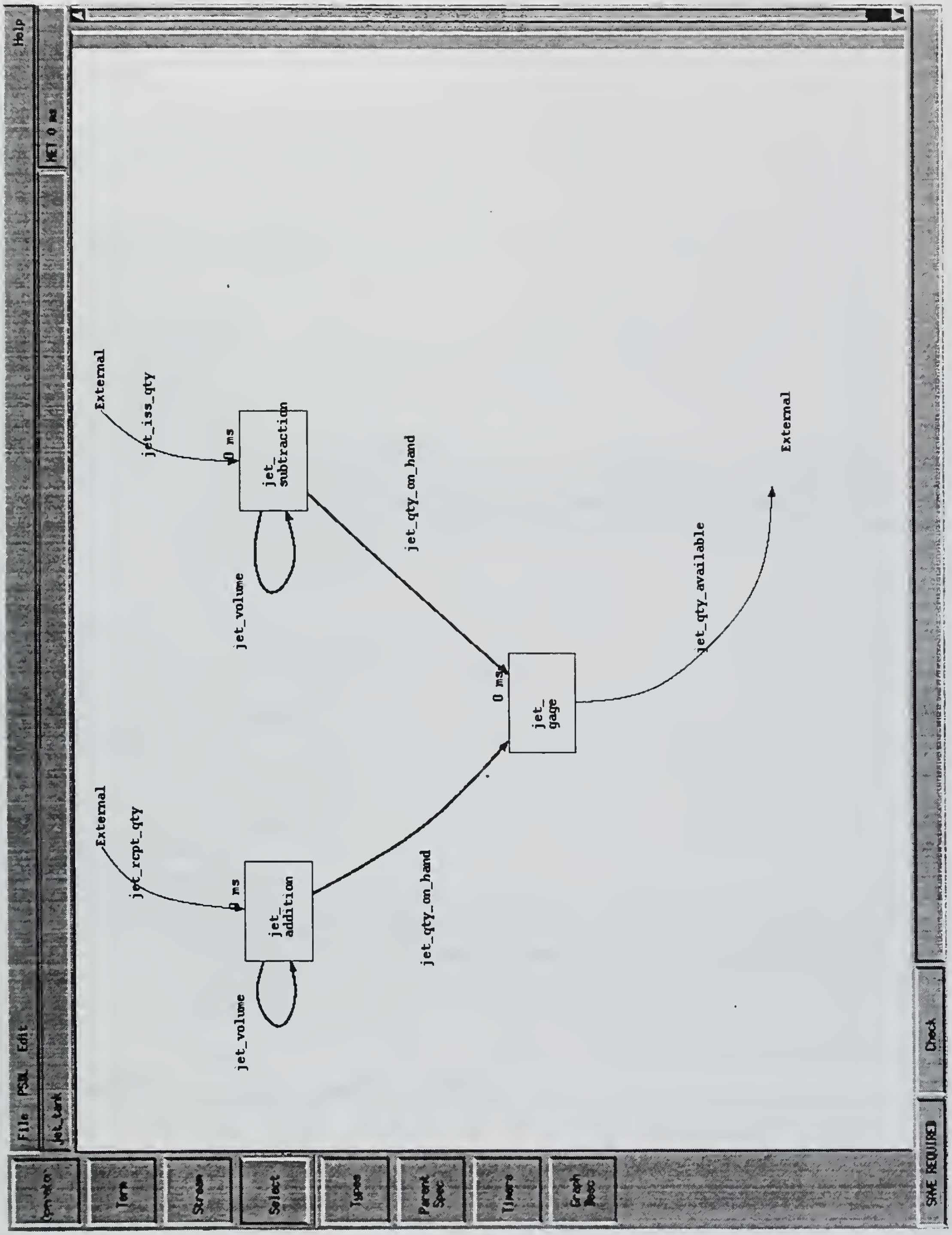


Figure 7. Decomposed Jet Fuel Tank

APPENDIX E

```
TYPE text_string
SPECIFICATION
END

IMPLEMENTATION  ADA text_string
END

OPERATOR other_iss_db_table_504
 SPECIFICATION
  INPUT
    eq_iss_name: text_string,
    eq_iss_unit: text_string,
    eq_iss_id: text_string,
    eq_iss_qty: integer,
    eq_iss_fuel_type: integer
  KEYWORDS table, database, equipment_issue
  DESCRIPTION {Table in a relational database to store the input parameters for all fuel issues to
                equipment. The table provides a historical audit trail of all equipment issues. Also
                provides a hook for future system enhancement such as data mining and statistical
                analysis applications.}
 END

 IMPLEMENTATION ADA other_iss_db_table_504
 END

OPERATOR other_rcpt_db_table_501
 SPECIFICATION
  INPUT
    oth_rcpt_source_unit: text_string,
    oth_rcpt_source_id: text_string,
    oth_rcpt_qty: integer,
    oth_rcpt_fuel_type: integer
  KEYWORDS table, database, other_receipt
  DESCRIPTION {Table in a relational database to store the input parameters of all receipts other than
                bulk. The table provides a historical audit trail of all receipts other than bulk. Also
                provides a hook for future system enhancement such as data mining and statistical
                analysis applications.}
 END

 IMPLEMENTATION ADA other_rcpt_db_table_501
 END

OPERATOR bulk_iss_db_table_498
 SPECIFICATION
  INPUT
    bulk_rcv_name: text_string,
    bulk_rcv_unit: text_string,
    bulk_iss_doc_num: text_string,
    bulk_iss_qty: integer,
```

```
    bulk_iss_fuel_type: integer
   KEYWORDS table, database, bulk_issue

   DESCRIPTION {Table in a relational database to store the input parameters of a bulk fuel issue. The
                table provides a historical audit trail of all bulk fuel issues. Also provides a hook for
                future system enhancements such as data mining and statistical analysis applications.}
  END

  IMPLEMENTATION ADA bulk_iss_db_table_498
  END

OPERATOR bulk_rcpt_db_table_495
  SPECIFICATION
   INPUT
    bulk_rcpt_doc_number: text_string,
    bulk_rcpt_qty: integer,
    bulk_rcpt_fuel_type: integer
   KEYWORDS table, database, bulk_receipt
   DESCRIPTION {Table in a relational database to store the input parameters of a bulk fuel receipt. The
                table provides a historical audit trail of all bulk fuel receipts. Also provides a hook for
                future system enhancement such as data mining and statistical analysis applications.}
  END

  IMPLEMENTATION ADA bulk_rcpt_db_table_495
  END

OPERATOR daily_reporter_410
  SPECIFICATION
   INPUT
    mg_iss_total: integer,
    df_iss_total: integer,
    jet_iss_total: integer,
    df_rcpt_total: integer,
    mg_rcpt_total: integer,
    jet_rcpt_total: integer
   OUTPUT
    daily_jet_iss_total: integer,
    daily_mg_iss_total: integer,
    daily_df_iss_total: integer,
    daily_jet_rcpt_total: integer,
    daily_mg_rcpt_total: integer,
    daily_df_rcpt_total: integer,
    mg_iss_total: integer,
    df_iss_total: integer,
    jet_iss_total: integer,
    jet_rcpt_total: integer,
    mg_rcpt_total: integer,
    df_rcpt_total: integer
   MAXIMUM EXECUTION TIME 750 MS
   KEYWORDS daily, periodic_operator
   DESCRIPTION {Periodic operator that forwards the daily total receipts and issues per fuel type every 1
                hour.  Note: The normal period of the daily reporter is 24 hours. It has reduced to 1
                hour for this prototype.}
  END

  IMPLEMENTATION ADA daily_reporter_410
```

```
 END

OPERATOR df_iss_totalizer_352
 SPECIFICATION
  INPUT
   i_df_qty: integer,
   df_iss_total: integer
  OUTPUT
   df_iss_total: integer
  KEYWORDS counter, diesel_fuel, issues
  DESCRIPTION {Counter of the daily quantity of diesel fuel issued. Also provides a hook for future
               system enhancement such as real time user views of the current daily total issues of
               diesel fuel on demand.}
 END

 IMPLEMENTATION ADA df_iss_totalizer_352
 END

OPERATOR mg_iss_totalizer_349
 SPECIFICATION
  INPUT
   i_mg_qty: integer,
   mg_iss_total: integer
  OUTPUT
   mg_iss_total: integer
  KEYWORDS counter, mogas, issues
  DESCRIPTION {Counter of the daily quantity of mogas issued. Also provides a hook for future system
               enhancement such as real time user views of the current daily total issues of mogas on
               demand.}
 END

 IMPLEMENTATION ADA mg_iss_totalizer_349
 END

OPERATOR jet_iss_totalizer_346
 SPECIFICATION
  INPUT
   i_jet_qty: integer,
   jet_iss_total: integer
  OUTPUT
   jet_iss_total: integer
  KEYWORDS counter, jet_fuel, issues
  DESCRIPTION {Counter of the daily quantity of jet fuel issued. Also provides a hook for future system
               enhancement such as real time user views of the current daily total issues of jet fuel on
               demand.}
 END

 IMPLEMENTATION ADA jet_iss_totalizer_346
 END

OPERATOR jet_rcpt_totalizer_280
 SPECIFICATION
  INPUT
   r_jet_qty: integer,
   jet_rcpt_total: integer
  OUTPUT
```

```
    jet_rcpt_total: integer
   KEYWORDS counter, jet_fuel, receipt
   DESCRIPTION {Counter of the daily quantity of jet fuel received. Also provides a hook for future
                system enhancement such as real time user views of the current daily total receipts of
                jet fuel on demand.}
  END

  IMPLEMENTATION ADA jet_rcpt_totalizer_280
  END

OPERATOR mg_rcpt_totalizer_277
  SPECIFICATION
   INPUT
    r_mg_qty: integer,
    mg_rcpt_total: integer
   OUTPUT
    mg_rcpt_total: integer
   KEYWORDS counter, mogas, receipt
   DESCRIPTION {Counter of the daily quantity of mogas received. Also provides a hook for future
                system enhancement such as real time user views of the current daily total receipts of
                mogas on demand.}
  END

  IMPLEMENTATION ADA mg_rcpt_totalizer_277
  END

OPERATOR df_rcpt_totalizer_274
  SPECIFICATION
   INPUT
    r_df_qty: integer,
    df_rcpt_total: integer
   OUTPUT
    df_rcpt_total: integer
   KEYWORDS counter, diesel_fuel, receipt
   DESCRIPTION {Counter of the daily quantity of diesel fuel received. Also provides a hook for future
                system enhancement such as real time user views of the current daily total receipts of
                diesel fuel on demand.}
  END

  IMPLEMENTATION ADA df_rcpt_totalizer_274
  END

OPERATOR bulk_rcpt_processor_198
  SPECIFICATION
   INPUT
    bulk_rcpt_fuel_type: integer,
    bulk_rcpt_qty: integer,
    oth_rcpt_enable: boolean
   OUTPUT
    bulk_rcpt_enable: boolean,
    oth_rcpt_enable: boolean
   KEYWORDS fuel_type, fuel_quantity, processor
   DESCRIPTION {Serves as a preprocessor. Insures the bulk receipt input parameters of type of fuel and
                quantity arrive to be processed at the same time by using a by all triggering condition.
                Arrival of both parameters cause an enable parameter to be generated indicating that
                the bulk receipt parameters are present and ready to be processed by the receipt
```

```
                    processor.}
 END

 IMPLEMENTATION ADA bulk_rcpt_processor_198
 END

OPERATOR oth_rcpt_processor_207
 SPECIFICATION
  INPUT
   oth_rcpt_qty: integer,
   oth_rcpt_fuel_type: integer,
   bulk_rcpt_enable: boolean
  OUTPUT
   oth_rcpt_enable: boolean,
   bulk_rcpt_enable: boolean
  KEYWORDS fuel_type, fuel_quantity, preprocessor
  DESCRIPTION {Serves as a preprocessor. Insures the other receipt input parameters of type of fuel and
                    quantity arrive to be processed at the same time using a by all triggering condition.
                    Arrival of both parameters cause an enable parameter to be generated indicating that
                    the other receipt parameters are present and ready to be processed by the receipt
                    processor.}
 END

 IMPLEMENTATION ADA oth_rcpt_processor_207
 END

OPERATOR rcpt_processor_210
 SPECIFICATION
  INPUT
   oth_rcpt_qty: integer,
   oth_rcpt_fuel_type: integer,
   bulk_rcpt_qty: integer,
   bulk_rcpt_fuel_type: integer,
   oth_rcpt_enable: boolean,
   bulk_rcpt_enable: boolean
  OUTPUT
   r_jet_qty: integer,
   r_mg_qty: integer,
   r_df_qty: integer,
   jet_rcpt_qty: integer,
   mogas_rcpt_qty: integer,
   diesel_rcpt_qty: integer
  KEYWORDS receipts, bulk_receipts, other_receipts, receipt_processor
  DESCRIPTION {Processor of all bulk and other fuel receipts. Based upon an enable signal and the type
                    of fuel, the processor passes the received quantity of fuel to the appropriate fuel
                    storage tank and totalizer.}
 END

 IMPLEMENTATION ADA rcpt_processor_210
 END

OPERATOR oth_iss_processor_307
 SPECIFICATION
  INPUT
   eq_iss_qty: integer,
   eq_iss_fuel_type: integer,
```

```
    bulk_iss_enable: boolean
   OUTPUT
    oth_iss_enable: boolean,
    bulk_iss_enable: boolean
   KEYWORDS fuel_type, fuel_quantity, preprocessor
   DESCRIPTION {Serves as a preprocessor. Insures the equipment issue input parameters of fuel type and
                quantity arrive to be processed at the same time by using a by all triggering condition.
                Arrival of both parameters cause an enable parameter to be generated indicating that
                the equipment issue parameters are present and ready to be processed by the issue
                processor.}
  END

  IMPLEMENTATION ADA oth_iss_processor_307
  END

OPERATOR bulk_iss_processor_310
  SPECIFICATION
   INPUT
    bulk_iss_qty: integer,
    bulk_iss_fuel_type: integer,
    oth_iss_enable: boolean
   OUTPUT
    bulk_iss_enable: boolean,
    oth_iss_enable: boolean
   KEYWORDS fuel_type, fuel_quantity, preprocessor
   DESCRIPTION {Serves as a preprocessor. Insures the bulk issue parameters of fuel type and quantity
                arrive to be processed at the same time by using a by all triggering condition. Arrival
                of both parameters cause an enable parameter to be generated indicating that the bulk
                issue parameters are present and ready to be processed by the issue processor.}
  END

  IMPLEMENTATION ADA bulk_iss_processor_310
  END

OPERATOR iss_processor_323
  SPECIFICATION
   INPUT
    bulk_iss_enable: boolean,
    oth_iss_enable: boolean,
    eq_iss_fuel_type: integer,
    eq_iss_qty: integer,
    bulk_iss_fuel_type: integer,
    bulk_iss_qty: integer
   OUTPUT
    i_df_qty: integer,
    i_mg_qty: integer,
    i_jet_qty: integer,
    diesel_iss_qty: integer,
    mogas_iss_qty: integer,
    jet_iss_qty: integer
   KEYWORDS issues, bulk_issues, equipment_issues, issue_processor
   DESCRIPTION {Processor of all bulk and equipment fuel issues. Based upon an enable signal and the
                type of fuel, the processor passes the issued quantity of fuel to the appropriate fuel
                storage tank and totalizer.}
  END
```

```
 IMPLEMENTATION ADA iss_processor_323
 END

OPERATOR jet_gage_917
 SPECIFICATION
  INPUT
   jet_qty_on_hand: integer
  OUTPUT
   jet_qty_available: integer
  KEYWORDS storage, tank, jet_fuel, gage
  DESCRIPTION {Simulates gaging the jet fuel storage tank to determine the quantity of fuel on hand.}
 END

 IMPLEMENTATION ADA jet_gage_917
 END

OPERATOR jet_subtraction_914
 SPECIFICATION
  INPUT
   jet_volume: integer,
   jet_iss_qty: integer
  OUTPUT
   jet_qty_on_hand: integer,
   jet_volume: integer
  KEYWORDS storage, tank, jet_fuel, issue
  DESCRIPTION {Simulates the issue/subtraction of a quantity of jet fuel from the storage tank.}
 END

 IMPLEMENTATION ADA jet_subtraction_914
 END

OPERATOR jet_addition_911
 SPECIFICATION
  INPUT
   jet_volume: integer,
   jet_rcpt_qty: integer
  OUTPUT
   jet_qty_on_hand: integer,
   jet_volume: integer
  KEYWORDS storage, tank, jet_fuel, receipt
  DESCRIPTION {Simulates the receipt/addition of a quantity of jet fuel to the storage tank.}
 END

 IMPLEMENTATION ADA jet_addition_911
 END

OPERATOR mogas_addition_888
 SPECIFICATION
  INPUT
   mogas_rcpt_qty: integer,
   mogas_volume: integer
  OUTPUT
   mogas_volume: integer,
   mg_qty_on_hand: integer
  KEYWORDS storage, tank, mogas, receipt
  DESCRIPTION {Simulates the receipt/addition of a quantity of mogas to the storage tank.}
```

```
  END

  IMPLEMENTATION ADA mogas_addition_888
  END

OPERATOR mogas_subtraction_891
  SPECIFICATION
    INPUT
      mogas_iss_qty: integer,
      mogas_volume: integer
    OUTPUT
      mogas_volume: integer,
      mg_qty_on_hand: integer
    KEYWORDS storage, tank, mogas, issue
    DESCRIPTION {Simulates the issue/subtraction of a quantity of mogas from the storage tank.}
  END

  IMPLEMENTATION ADA mogas_subtraction_891
  END

OPERATOR mogas_gage_894
  SPECIFICATION
    INPUT
      mg_qty_on_hand: integer
    OUTPUT
      mogas_qty_available: integer
    KEYWORDS storage, tank, mogas, gage
    DESCRIPTION {Simulates the gaging of the mogas storage tank to determine the quantity of fuel on
hand.}
  END

  IMPLEMENTATION ADA mogas_gage_894
  END

OPERATOR gui_bulk_receipt_3
  SPECIFICATION
    OUTPUT
      bulk_rcpt_fuel_type: integer,
      bulk_rcpt_doc_number: text_string,
      bulk_rcpt_qty: integer
    KEYWORDS bulk_receipt, user_input, dd_form_1348_1, gui
    DESCRIPTION {Allows a petroleum specialist to manually input parameters extracted from DD Form
                1348-1 during the receipt of bulk petroleum.  The petroleum specialist enters the type
                of fuel received, quantity in gallons, and the document number as required by Army
                Regulation 710-2.}
  END

  IMPLEMENTATION ADA gui_bulk_receipt_3
  END

OPERATOR gui_other_receipt_6
  SPECIFICATION
    OUTPUT
      oth_rcpt_qty: integer,
      oth_rcpt_source_unit: text_string,
      oth_rcpt_fuel_type: integer,
```

```
      oth_rcpt_source_id: text_string
    KEYWORDS gui, other_receipt, user_input
    DESCRIPTION {Interface capturing all other possible petroleum receipt scenarios i.e. vehicle/aircraft
                 defueling, etc.  The interface allows a petroleum specialist to manually enter the type
                 of fuel received, quantity in gallons, an identification number from the source, and
                 source unit as required by Army Regulation 710-2.}
  END

  IMPLEMENTATION ADA gui_other_receipt_6
  END

OPERATOR gui_bulk_issue_9
  SPECIFICATION
    OUTPUT
      bulk_rcv_unit: text_string,
      bulk_iss_doc_num: text_string,
      bulk_rcv_name: text_string,
      bulk_iss_qty: integer,
      bulk_iss_fuel_type: integer
    KEYWORDS gui, bulk_issue, da_form_2765_1, user_input
    DESCRIPTION {Interface that allows a petroleum specialist to manually input parameters extracted
                 from DA Form 2765-1 during issue of bulk petroleum.  The petroleum specialist enters
                 the type of fuel issued, quantity in gallons, document number of the issue, the
                 receiving unit, and the name/rank of the receiver as required by Army
                 Regulation 710-2.}
  END

  IMPLEMENTATION ADA gui_bulk_issue_9
  END

OPERATOR gui_other_issue_12
  SPECIFICATION
    OUTPUT
      eq_iss_unit: text_string,
      eq_iss_name: text_string,
      eq_iss_fuel_type: integer,
      eq_iss_id: text_string,
      eq_iss_qty: integer
    KEYWORDS gui, user_input, other_issue
    DESCRIPTION {Interface that allows a petroleum specialist to manually input parameters required by
                 Army Regulation 710-2 for all fuel issues other than bulk issues. These other issues
                 include fuel issues made directly into or specifically identifiable to a consuming end
                 item. An example of this type of issue is to a vehicle or a M2 burner unit. Petroleum
                 specialist enters the type of fuel issued, quantity in gallons, the receiving vehicle
                 bumper number/equipment name, the receiving unit, and the name/rank of
                 the receiver.}
  END

  IMPLEMENTATION ADA gui_other_issue_12
  END

OPERATOR daily_iss_db_table_743
  SPECIFICATION
    INPUT
      daily_df_iss_total: integer,
      daily_mg_iss_total: integer,
```

```
    daily_jet_iss_total: integer
  KEYWORDS table, data_base, issue
  DESCRIPTION {Table in a relational database to store the total daily issues per fuel type. The table
               provides a historical audit trail. Also provides a hook for future system enhancement
               such as data mining and statistical analysis applications.}
 END

 IMPLEMENTATION ADA daily_iss_db_table_743
 END

OPERATOR daily_rcpt_db_table_740
 SPECIFICATION
  INPUT
    daily_df_rcpt_total: integer,
    daily_mg_rcpt_total: integer,
    daily_jet_rcpt_total: integer
  KEYWORDS table, database, receipt
  DESCRIPTION {Table in a relational database to store the daily total receipts per fuel type.  The table
               provides a historical audit trail.  Also provides a hook for future system enhancement
               such as data mining and statistical analysis applications.}
 END

 IMPLEMENTATION ADA daily_rcpt_db_table_740
 END

OPERATOR monthly_reporter_601
 SPECIFICATION
  INPUT
    mo_iss_jet_total: integer,
    mo_iss_mg_total: integer,
    mo_iss_df_total: integer,
    mo_rcpt_jet_total: integer,
    mo_rcpt_mg_total: integer,
    mo_rcpt_df_total: integer
  OUTPUT
    mo_iss_jet_total: integer,
    mo_iss_mg_total: integer,
    mo_iss_df_total: integer,
    mo_rcpt_jet_total: integer,
    mo_rcpt_mg_total: integer,
    mo_rcpt_df_total: integer,
    month_df_iss_total: integer,
    month_df_rcpt_total: integer,
    month_mg_iss_total: integer,
    month_mg_rcpt_total: integer,
    month_jet_rcpt_total: integer,
    month_jet_iss_total: integer
  MAXIMUM EXECUTION TIME 750 MS
  KEYWORDS monthly, periodic_operator
  DESCRIPTION {Periodic operator that forwards the monthly total receipts and issues per fuel type
               every 30 hours.  Note: The normal period of the monthly reporter is 720 hours,
               approximately one month. It has been proportionately reduced to 30 hours for this
               prototype.}
 END

 IMPLEMENTATION ADA monthly_reporter_601
```

```
 END

OPERATOR mo_jet_iss_totalizer_598
 SPECIFICATION
  INPUT
   mo_iss_jet_total: integer,
   daily_jet_iss_total: integer
  OUTPUT
   mo_iss_jet_total: integer
  KEYWORDS counter, jet_fuel, issue
  DESCRIPTION {Counter of the quantity of jet fuel issued over the course of the month. Also provides a
               hook for future system enhancement such as user views of the current total issues of jet
               fuel for the month.}
 END

 IMPLEMENTATION ADA mo_jet_iss_totalizer_598
 END

OPERATOR mo_mg_iss_totalizer_595
 SPECIFICATION
  INPUT
   mo_iss_mg_total: integer,
   daily_mg_iss_total: integer
  OUTPUT
   mo_iss_mg_total: integer
  KEYWORDS counter, mogas, issue
  DESCRIPTION {Counter of the quantity of mogas issued over the course of the month. Also provides a
               hook for future system enhancement such as user views of the current total issues of
               mogas for the month.}
 END

 IMPLEMENTATION ADA mo_mg_iss_totalizer_595
 END

OPERATOR mo_df_iss_totalizer_592
 SPECIFICATION
  INPUT
   mo_iss_df_total: integer,
   daily_df_iss_total: integer
  OUTPUT
   mo_iss_df_total: integer
  KEYWORDS counter, diesel_fuel, issue
  DESCRIPTION {Counter of the quantity of diesel fuel issued over the course of the month. Also
               provides a hook for future system enhancement such as user views of the current total
               issues of diesel fuel for the month.}
 END

 IMPLEMENTATION ADA mo_df_iss_totalizer_592
 END

OPERATOR mo_jet_rcpt_totalizer_589
 SPECIFICATION
  INPUT
   daily_jet_rcpt_total: integer,
   mo_rcpt_jet_total: integer
  OUTPUT
```

```
    mo_rcpt_jet_total: integer
   KEYWORDS counter, jet_fuel, receipt
   DESCRIPTION {Counter of the quantity of jet fuel received over the course of the month. Also provides
                a hook for future system enhancement such as user views of the current total receipts
                of jet fuel for the month.}
  END

  IMPLEMENTATION ADA mo_jet_rcpt_totalizer_589
  END

OPERATOR mo_mg_rcpt_totalizer_586
  SPECIFICATION
   INPUT
    daily_mg_rcpt_total: integer,
    mo_rcpt_mg_total: integer
   OUTPUT
    mo_rcpt_mg_total: integer
   KEYWORDS counter, mogas, receipt

   DESCRIPTION {Counter of the quantity of mogas received over the course of the month. Also provides
                a hook for future system enhancement such as user views of the current total receipts
                of mogas for the month.}
  END

  IMPLEMENTATION ADA mo_mg_rcpt_totalizer_586
  END

OPERATOR mo_df_rcpt_totalizer_583
  SPECIFICATION
   INPUT
    daily_df_rcpt_total: integer,
    mo_rcpt_df_total: integer
   OUTPUT
    mo_rcpt_df_total: integer
   KEYWORDS counter, diesel_fuel, receipt
   DESCRIPTION {Counter of the quantity of diesel fuel received over the course of the month. Also
                provides a hook for future system enhancement such as user views of the current total
                receipts of diesel fuel for the month.}
  END

  IMPLEMENTATION ADA mo_df_rcpt_totalizer_583
  END

OPERATOR diesel_iss_acct_proc_934
  SPECIFICATION
   INPUT
    month_df_iss_total: integer
   OUTPUT
    total_mo_df_iss: integer
   KEYWORDS diesel, issue, accountability
   DESCRIPTION {Sets the total monthly diesel issues for processing.}
  END

  IMPLEMENTATION ADA diesel_iss_acct_proc_934
  END
```

```
OPERATOR diesel_rcpt_acct_proc_937
 SPECIFICATION
  INPUT
   month_df_rcpt_total: integer
  OUTPUT
   total_mo_df_rcpt: integer
  KEYWORDS diesel, accountability, receipt
  DESCRIPTION {Sets the total monthly diesel receipts for processing.}
 END

 IMPLEMENTATION ADA diesel_rcpt_acct_proc_937
 END

OPERATOR mogas_iss_acct_proc_940
 SPECIFICATION
  INPUT
   month_mg_iss_total: integer
  OUTPUT
   total_mo_mg_iss: integer
  KEYWORDS mogas, accountability, issues
  DESCRIPTION {Sets the total monthly mogas issues for processing.}
 END

 IMPLEMENTATION ADA mogas_iss_acct_proc_940
 END

OPERATOR mogas_rcpt_acct_proc_943
 SPECIFICATION
  INPUT
   month_mg_rcpt_total: integer
  OUTPUT
   total_mo_mg_rcpt: integer
  KEYWORDS mogas, accountability, receipt
  DESCRIPTION {Sets the total monthly mogas receipts for processing.}
 END

 IMPLEMENTATION ADA mogas_rcpt_acct_proc_943
 END

OPERATOR jet_iss_acct_proc_946
 SPECIFICATION
  INPUT
   month_jet_iss_total: integer
  OUTPUT
   total_mo_jet_iss: integer
  KEYWORDS jet_fuel, accountability, issue
  DESCRIPTION {Sets the total monthly jet issues for processing.}
 END

 IMPLEMENTATION ADA jet_iss_acct_proc_946
 END

OPERATOR jet_rcpt_acct_proc_949
 SPECIFICATION
  INPUT
   month_jet_rcpt_total: integer
```

```
  OUTPUT
   total_mo_jet_rcpt: integer
  KEYWORDS jet_fuel, accountability, receipt
  DESCRIPTION {Sets the total monthly jet receipts for processing.}
 END

 IMPLEMENTATION ADA jet_rcpt_acct_proc_949
 END

OPERATOR df_acct_calc_952
 SPECIFICATION
  INPUT
   total_mo_df_iss: integer,
   total_mo_df_rcpt: integer,
   opening_inv_diesel: integer,
   diesel_qty_available: integer
  OUTPUT
   tolerance_df: boolean,
   opening_inv_diesel: integer
  KEYWORDS diesel, tolerance, accountability
  DESCRIPTION {Determines whether diesel fuel accountability is within tolerance.}
 END

 IMPLEMENTATION ADA df_acct_calc_952
 END

OPERATOR mg_acct_calc_955
 SPECIFICATION
  INPUT
   total_mo_mg_iss: integer,
   total_mo_mg_rcpt: integer,
   opening_inv_mogas: integer,
   mogas_qty_available: integer
  OUTPUT
   tolerance_mg: boolean,
   opening_inv_mogas: integer
  KEYWORDS mogas, tolerance, accountability
  DESCRIPTION {Determines whether mogas fuel accountability is within tolerance.}
 END

 IMPLEMENTATION ADA mg_acct_calc_955
 END

OPERATOR jet_acct_calc_958
 SPECIFICATION
  INPUT
   total_mo_jet_iss: integer,
   total_mo_jet_rcpt: integer,
   opening_inv_jet: integer,
   jet_qty_available: integer
  OUTPUT
   tolerance_jet: boolean,
   opening_inv_jet: integer
  KEYWORDS jet_fuel, tolerance, accountability
  DESCRIPTION {Determines whether jet fuel accountability is within tolerance.}
 END
```

```
 IMPLEMENTATION ADA jet_acct_calc_958
 END

OPERATOR gui_fuel_on_hand_124
 SPECIFICATION
  INPUT
   jet_qty_available: integer,
   mogas_qty_available: integer,
   diesel_qty_available: integer
  KEYWORDS gui, fuel_balance_on_hand, jet_fuel_balance, diesel_fuel_balance, mogas_balance
  DESCRIPTION {Interface that shows the current fuel totals in gallons that are stored and available for
               issue.  This interface is available to both the using petroleum specialist and
               accountable officer.}
 END

 IMPLEMENTATION ADA gui_fuel_on_hand_124
 END

OPERATOR gui_acc_officer_179
 SPECIFICATION
  INPUT
   tolerance_df: boolean,
   tolerance_mg: boolean,
   tolerance_jet: boolean
  KEYWORDS gui, accountability_report, fuel_tolerance
  DESCRIPTION {Interface for the accountable officer to view the monthly fuel report.}
 END

 IMPLEMENTATION ADA gui_acc_officer_179
 END

OPERATOR diesel_gage_854
 SPECIFICATION
  INPUT
   df_qty_on_hand: integer
  OUTPUT
   diesel_qty_available: integer
  KEYWORDS storage, tank, diesel, gage
  DESCRIPTION {Simulates gaging the diesel storage tank to determine the quantity of fuel on hand.}
 END

 IMPLEMENTATION ADA diesel_gage_854
 END

OPERATOR diesel_subtraction_839
 SPECIFICATION
  INPUT
   diesel_volume: integer,
   diesel_iss_qty: integer
  OUTPUT
   df_qty_on_hand: integer,
   diesel_volume: integer
  KEYWORDS storage, tank, diesel, issue
  DESCRIPTION {Simulates the issue/subtraction of a quantity of diesel fuel from the storage tank.}
 END
```

```
 IMPLEMENTATION ADA diesel_subtraction_839
 END

OPERATOR diesel_addition_836
 SPECIFICATION
  INPUT
   diesel_volume: integer,
   diesel_rcpt_qty: integer
  OUTPUT
   df_qty_on_hand: integer,
   diesel_volume: integer
  KEYWORDS storage, tank, diesel, receipt
  DESCRIPTION {Simulates the receipt/addition of a quantity of diesel fuel to the storage tank.}
 END

 IMPLEMENTATION ADA diesel_addition_836
 END

OPERATOR fuel_subsystem_1
 SPECIFICATION
  STATES jet_qty_on_hand: integer INITIALLY 0
  STATES jet_volume: integer INITIALLY 0
  STATES mogas_volume: integer INITIALLY 0
  STATES mg_qty_on_hand: integer INITIALLY 0
  STATES jet_iss_total: integer INITIALLY 0
  STATES mg_iss_total: integer INITIALLY 0
  STATES df_iss_total: integer INITIALLY 0
  STATES jet_rcpt_total: integer INITIALLY 0
  STATES mg_rcpt_total: integer INITIALLY 0
  STATES df_rcpt_total: integer INITIALLY 0
  STATES oth_rcpt_enable: boolean INITIALLY FALSE
  STATES bulk_rcpt_enable: boolean INITIALLY FALSE
  STATES bulk_iss_enable: boolean INITIALLY FALSE
  STATES oth_iss_enable: boolean INITIALLY FALSE
  STATES mo_iss_jet_total: integer INITIALLY 0
  STATES mo_iss_mg_total: integer INITIALLY 0
  STATES mo_iss_df_total: integer INITIALLY 0
  STATES mo_rcpt_jet_total: integer INITIALLY 0
  STATES mo_rcpt_mg_total: integer INITIALLY 0
  STATES mo_rcpt_df_total: integer INITIALLY 0
  STATES total_mo_df_iss: integer INITIALLY 0
  STATES total_mo_df_rcpt: integer INITIALLY 0
  STATES total_mo_mg_iss: integer INITIALLY 0
  STATES total_mo_mg_rcpt: integer INITIALLY 0
  STATES total_mo_jet_iss: integer INITIALLY 0
  STATES total_mo_jet_rcpt: integer INITIALLY 0
  STATES opening_inv_diesel: integer INITIALLY 0
  STATES opening_inv_mogas: integer INITIALLY 0
  STATES opening_inv_jet: integer INITIALLY 0
  STATES df_qty_on_hand: integer INITIALLY 0
  STATES diesel_volume: integer INITIALLY 0
 END

 IMPLEMENTATION
  GRAPH
```

VERTEX gui_bulk_receipt_3
VERTEX gui_other_receipt_6
VERTEX gui_bulk_issue_9
VERTEX gui_other_issue_12
VERTEX gui_fuel_on_hand_124
VERTEX gui_acc_officer_179
VERTEX jet_gage_917
VERTEX jet_subtraction_914
VERTEX jet_addition_911
VERTEX mogas_addition_888
VERTEX mogas_subtraction_891
VERTEX mogas_gage_894
VERTEX other_iss_db_table_504
VERTEX other_rcpt_db_table_501
VERTEX bulk_iss_db_table_498
VERTEX bulk_rcpt_db_table_495
VERTEX daily_reporter_410: 750 MS
VERTEX df_iss_totalizer_352
VERTEX mg_iss_totalizer_349
VERTEX jet_iss_totalizer_346
VERTEX jet_rcpt_totalizer_280
VERTEX mg_rcpt_totalizer_277
VERTEX df_rcpt_totalizer_274
VERTEX bulk_rcpt_processor_198
VERTEX oth_rcpt_processor_207
VERTEX rcpt_processor_210
VERTEX oth_iss_processor_307
VERTEX bulk_iss_processor_310
VERTEX iss_processor_323
VERTEX daily_iss_db_table_743
VERTEX daily_rcpt_db_table_740
VERTEX monthly_reporter_601: 750 MS
VERTEX mo_jet_iss_totalizer_598
VERTEX mo_mg_iss_totalizer_595
VERTEX mo_df_iss_totalizer_592
VERTEX mo_jet_rcpt_totalizer_589
VERTEX mo_mg_rcpt_totalizer_586
VERTEX mo_df_rcpt_totalizer_583
VERTEX diesel_iss_acct_proc_934
VERTEX diesel_rcpt_acct_proc_937
VERTEX mogas_iss_acct_proc_940
VERTEX mogas_rcpt_acct_proc_943
VERTEX jet_iss_acct_proc_946
VERTEX jet_rcpt_acct_proc_949
VERTEX df_acct_calc_952
VERTEX mg_acct_calc_955
VERTEX jet_acct_calc_958
VERTEX diesel_gage_854
VERTEX diesel_subtraction_839
VERTEX diesel_addition_836

EDGE jet_qty_on_hand jet_subtraction_914 -> jet_gage_917
EDGE jet_qty_on_hand jet_addition_911 -> jet_gage_917
EDGE jet_volume jet_addition_911 -> jet_addition_911
EDGE jet_volume jet_subtraction_914 -> jet_subtraction_914
EDGE jet_qty_available jet_gage_917 -> gui_fuel_on_hand_124

```
EDGE mogas_volume mogas_addition_888 -> mogas_addition_888
EDGE mogas_volume mogas_subtraction_891 -> mogas_subtraction_891
EDGE mg_qty_on_hand mogas_addition_888 -> mogas_gage_894
EDGE mg_qty_on_hand mogas_subtraction_891 -> mogas_gage_894
EDGE mogas_qty_available mogas_gage_894 -> gui_fuel_on_hand_124
EDGE jet_iss_total jet_iss_totalizer_346 -> daily_reporter_410
EDGE mg_iss_total mg_iss_totalizer_349 -> daily_reporter_410
EDGE df_iss_total df_iss_totalizer_352 -> daily_reporter_410
EDGE mg_iss_total daily_reporter_410 -> daily_reporter_410
EDGE df_iss_total daily_reporter_410 -> daily_reporter_410
EDGE jet_iss_total daily_reporter_410 -> daily_reporter_410
EDGE jet_rcpt_total daily_reporter_410 -> daily_reporter_410
EDGE mg_rcpt_total daily_reporter_410 -> daily_reporter_410
EDGE df_rcpt_total daily_reporter_410 -> daily_reporter_410
EDGE df_rcpt_total df_rcpt_totalizer_274 -> daily_reporter_410
EDGE mg_rcpt_total mg_rcpt_totalizer_277 -> daily_reporter_410
EDGE jet_rcpt_total jet_rcpt_totalizer_280 -> daily_reporter_410
EDGE i_df_qty iss_processor_323 -> df_iss_totalizer_352
EDGE i_mg_qty iss_processor_323 -> mg_iss_totalizer_349
EDGE i_jet_qty iss_processor_323 -> jet_iss_totalizer_346
EDGE r_jet_qty rcpt_processor_210 -> jet_rcpt_totalizer_280
EDGE r_mg_qty rcpt_processor_210 -> mg_rcpt_totalizer_277
EDGE r_df_qty rcpt_processor_210 -> df_rcpt_totalizer_274
EDGE oth_rcpt_enable oth_rcpt_processor_207 -> rcpt_processor_210
EDGE bulk_rcpt_enable bulk_rcpt_processor_198 -> rcpt_processor_210
EDGE bulk_iss_enable bulk_iss_processor_310 -> iss_processor_323
EDGE oth_iss_enable oth_iss_processor_307 -> iss_processor_323
EDGE df_rcpt_total df_rcpt_totalizer_274 -> df_rcpt_totalizer_274
EDGE mg_rcpt_total mg_rcpt_totalizer_277 -> mg_rcpt_totalizer_277
EDGE jet_rcpt_total jet_rcpt_totalizer_280 -> jet_rcpt_totalizer_280
EDGE jet_iss_total jet_iss_totalizer_346 -> jet_iss_totalizer_346
EDGE df_iss_total df_iss_totalizer_352 -> df_iss_totalizer_352
EDGE mg_iss_total mg_iss_totalizer_349 -> mg_iss_totalizer_349
EDGE bulk_rcpt_enable oth_rcpt_processor_207 -> oth_rcpt_processor_207
EDGE oth_rcpt_enable bulk_rcpt_processor_198 -> bulk_rcpt_processor_198
EDGE oth_iss_enable bulk_iss_processor_310 -> bulk_iss_processor_310
EDGE bulk_iss_enable oth_iss_processor_307 -> oth_iss_processor_307
EDGE eq_iss_unit gui_other_issue_12 -> other_iss_db_table_504
EDGE eq_iss_name gui_other_issue_12 -> other_iss_db_table_504
EDGE eq_iss_fuel_type gui_other_issue_12 -> other_iss_db_table_504
EDGE eq_iss_fuel_type gui_other_issue_12 -> oth_iss_processor_307
EDGE eq_iss_fuel_type gui_other_issue_12 -> iss_processor_323
EDGE eq_iss_id gui_other_issue_12 -> other_iss_db_table_504
EDGE eq_iss_qty gui_other_issue_12 -> other_iss_db_table_504
EDGE eq_iss_qty gui_other_issue_12 -> oth_iss_processor_307
EDGE eq_iss_qty gui_other_issue_12 -> iss_processor_323
EDGE bulk_rcv_unit gui_bulk_issue_9 -> bulk_iss_db_table_498
EDGE bulk_iss_doc_num gui_bulk_issue_9 -> bulk_iss_db_table_498
EDGE bulk_rcv_name gui_bulk_issue_9 -> bulk_iss_db_table_498
EDGE bulk_iss_qty gui_bulk_issue_9 -> bulk_iss_db_table_498
EDGE bulk_iss_qty gui_bulk_issue_9 -> bulk_iss_processor_310
EDGE bulk_iss_qty gui_bulk_issue_9 -> iss_processor_323
EDGE bulk_iss_fuel_type gui_bulk_issue_9 -> bulk_iss_db_table_498
EDGE bulk_iss_fuel_type gui_bulk_issue_9 -> bulk_iss_processor_310
EDGE bulk_iss_fuel_type gui_bulk_issue_9 -> iss_processor_323
EDGE oth_rcpt_qty gui_other_receipt_6 -> other_rcpt_db_table_501
```

EDGE oth_rcpt_qty gui_other_receipt_6 -> oth_rcpt_processor_207
EDGE oth_rcpt_qty gui_other_receipt_6 -> rcpt_processor_210
EDGE oth_rcpt_source_unit gui_other_receipt_6 -> other_rcpt_db_table_501
EDGE oth_rcpt_fuel_type gui_other_receipt_6 -> other_rcpt_db_table_501
EDGE oth_rcpt_fuel_type gui_other_receipt_6 -> oth_rcpt_processor_207
EDGE oth_rcpt_fuel_type gui_other_receipt_6 -> rcpt_processor_210
EDGE oth_rcpt_source_id gui_other_receipt_6 -> other_rcpt_db_table_501
EDGE bulk_rcpt_fuel_type gui_bulk_receipt_3 -> bulk_rcpt_db_table_495
EDGE bulk_rcpt_fuel_type gui_bulk_receipt_3 -> bulk_rcpt_processor_198
EDGE bulk_rcpt_fuel_type gui_bulk_receipt_3 -> rcpt_processor_210
EDGE bulk_rcpt_doc_number gui_bulk_receipt_3 -> bulk_rcpt_db_table_495
EDGE bulk_rcpt_qty gui_bulk_receipt_3 -> bulk_rcpt_db_table_495
EDGE bulk_rcpt_qty gui_bulk_receipt_3 -> bulk_rcpt_processor_198
EDGE bulk_rcpt_qty gui_bulk_receipt_3 -> rcpt_processor_210
EDGE jet_iss_qty iss_processor_323 -> jet_subtraction_914
EDGE jet_rcpt_qty rcpt_processor_210 -> jet_addition_911
EDGE mogas_rcpt_qty rcpt_processor_210 -> mogas_addition_888
EDGE mogas_iss_qty iss_processor_323 -> mogas_subtraction_891
EDGE mo_iss_jet_total monthly_reporter_601 -> monthly_reporter_601
EDGE mo_iss_mg_total monthly_reporter_601 -> monthly_reporter_601
EDGE mo_iss_df_total monthly_reporter_601 -> monthly_reporter_601
EDGE mo_rcpt_jet_total monthly_reporter_601 -> monthly_reporter_601
EDGE mo_rcpt_mg_total monthly_reporter_601 -> monthly_reporter_601
EDGE mo_rcpt_df_total monthly_reporter_601 -> monthly_reporter_601
EDGE mo_iss_jet_total mo_jet_iss_totalizer_598 -> monthly_reporter_601
EDGE mo_iss_jet_total mo_jet_iss_totalizer_598 -> mo_jet_iss_totalizer_598
EDGE mo_iss_mg_total mo_mg_iss_totalizer_595 -> monthly_reporter_601
EDGE mo_iss_mg_total mo_mg_iss_totalizer_595 -> mo_mg_iss_totalizer_595
EDGE mo_iss_df_total mo_df_iss_totalizer_592 -> monthly_reporter_601
EDGE mo_iss_df_total mo_df_iss_totalizer_592 -> mo_df_iss_totalizer_592
EDGE mo_rcpt_jet_total mo_jet_rcpt_totalizer_589 -> monthly_reporter_601
EDGE mo_rcpt_jet_total mo_jet_rcpt_totalizer_589 -> mo_jet_rcpt_totalizer_589
EDGE mo_rcpt_mg_total mo_mg_rcpt_totalizer_586 -> monthly_reporter_601
EDGE mo_rcpt_mg_total mo_mg_rcpt_totalizer_586 -> mo_mg_rcpt_totalizer_586
EDGE mo_rcpt_df_total mo_df_rcpt_totalizer_583 -> mo_df_rcpt_totalizer_583
EDGE mo_rcpt_df_total mo_df_rcpt_totalizer_583 -> monthly_reporter_601
EDGE daily_df_rcpt_total daily_reporter_410 -> daily_rcpt_db_table_740
EDGE daily_df_rcpt_total daily_reporter_410 -> mo_df_rcpt_totalizer_583
EDGE daily_df_iss_total daily_reporter_410 -> daily_iss_db_table_743
EDGE daily_df_iss_total daily_reporter_410 -> mo_df_iss_totalizer_592
EDGE daily_jet_iss_total daily_reporter_410 -> daily_iss_db_table_743
EDGE daily_jet_iss_total daily_reporter_410 -> mo_jet_iss_totalizer_598
EDGE daily_jet_rcpt_total daily_reporter_410 -> daily_rcpt_db_table_740
EDGE daily_jet_rcpt_total daily_reporter_410 -> mo_jet_rcpt_totalizer_589
EDGE daily_mg_iss_total daily_reporter_410 -> daily_iss_db_table_743
EDGE daily_mg_iss_total daily_reporter_410 -> mo_mg_iss_totalizer_595
EDGE daily_mg_rcpt_total daily_reporter_410 -> daily_rcpt_db_table_740
EDGE daily_mg_rcpt_total daily_reporter_410 -> mo_mg_rcpt_totalizer_586
EDGE total_mo_df_iss diesel_iss_acct_proc_934 -> df_acct_calc_952
EDGE total_mo_df_rcpt diesel_rcpt_acct_proc_937 -> df_acct_calc_952
EDGE total_mo_mg_iss mogas_iss_acct_proc_940 -> mg_acct_calc_955
EDGE total_mo_mg_rcpt mogas_rcpt_acct_proc_943 -> mg_acct_calc_955
EDGE total_mo_jet_iss jet_iss_acct_proc_946 -> jet_acct_calc_958
EDGE total_mo_jet_rcpt jet_rcpt_acct_proc_949 -> jet_acct_calc_958
EDGE opening_inv_diesel df_acct_calc_952 -> df_acct_calc_952
EDGE opening_inv_mogas mg_acct_calc_955 -> mg_acct_calc_955

EDGE opening_inv_jet jet_acct_calc_958 -> jet_acct_calc_958
EDGE jet_qty_available jet_gage_917 -> jet_acct_calc_958
EDGE mogas_qty_available mogas_gage_894 -> mg_acct_calc_955
EDGE month_df_iss_total monthly_reporter_601 -> diesel_iss_acct_proc_934
EDGE month_df_rcpt_total monthly_reporter_601 -> diesel_rcpt_acct_proc_937
EDGE month_jet_iss_total monthly_reporter_601 -> jet_iss_acct_proc_946
EDGE month_jet_rcpt_total monthly_reporter_601 -> jet_rcpt_acct_proc_949
EDGE month_mg_iss_total monthly_reporter_601 -> mogas_iss_acct_proc_940
EDGE month_mg_rcpt_total monthly_reporter_601 -> mogas_rcpt_acct_proc_943
EDGE tolerance_df df_acct_calc_952 -> gui_acc_officer_179
EDGE tolerance_mg mg_acct_calc_955 -> gui_acc_officer_179
EDGE tolerance_jet jet_acct_calc_958 -> gui_acc_officer_179
EDGE df_qty_on_hand diesel_subtraction_839 -> diesel_gage_854
EDGE df_qty_on_hand diesel_addition_836 -> diesel_gage_854
EDGE diesel_volume diesel_subtraction_839 -> diesel_subtraction_839
EDGE diesel_volume diesel_addition_836 -> diesel_addition_836
EDGE diesel_rcpt_qty rcpt_processor_210 -> diesel_addition_836
EDGE diesel_iss_qty iss_processor_323 -> diesel_subtraction_839
EDGE diesel_qty_available diesel_gage_854 -> gui_fuel_on_hand_124
EDGE diesel_qty_available diesel_gage_854 -> df_acct_calc_952

DATA STREAM
jet_qty_available: integer,
mogas_qty_available: integer,
jet_iss_qty: integer,
jet_rcpt_qty: integer,
mogas_rcpt_qty: integer,
mogas_iss_qty: integer,
diesel_qty_available: integer,
month_df_iss_total: integer,
month_df_rcpt_total: integer,
daily_df_rcpt_total: integer,
daily_df_iss_total: integer,
eq_iss_unit: text_string,
eq_iss_name: text_string,
eq_iss_fuel_type: integer,
eq_iss_id: text_string,
eq_iss_qty: integer,
bulk_rcv_unit: text_string,
bulk_iss_doc_num: text_string,
bulk_rcv_name: text_string,
bulk_iss_qty: integer,
bulk_iss_fuel_type: integer,
oth_rcpt_qty: integer,
oth_rcpt_source_unit: text_string,
oth_rcpt_fuel_type: integer,
oth_rcpt_source_id: text_string,
bulk_rcpt_fuel_type: integer,
bulk_rcpt_doc_number: text_string,
bulk_rcpt_qty: integer,
daily_jet_iss_total: integer,
daily_jet_rcpt_total: integer,
daily_mg_iss_total: integer,
daily_mg_rcpt_total: integer,
month_jet_iss_total: integer,
month_jet_rcpt_total: integer,

```
      month_mg_iss_total: integer,
      month_mg_rcpt_total: integer,
      tolerance_df: boolean,
      tolerance_mg: boolean,
      tolerance_jet: boolean,
      diesel_rcpt_qty: integer,
      diesel_iss_qty: integer,
      i_df_qty: integer,
      i_mg_qty: integer,
      i_jet_qty: integer,
      r_jet_qty: integer,
      r_mg_qty: integer,
      r_df_qty: integer
    CONTROL CONSTRAINTS
      OPERATOR gui_bulk_receipt_3
      OPERATOR gui_other_receipt_6
      OPERATOR gui_bulk_issue_9
      OPERATOR gui_other_issue_12
      OPERATOR gui_fuel_on_hand_124
      OPERATOR gui_acc_officer_179
      OPERATOR jet_gage_917
        TRIGGERED BY SOME jet_qty_on_hand
      OPERATOR jet_subtraction_914
        TRIGGERED BY ALL jet_iss_qty
      OPERATOR jet_addition_911
        TRIGGERED BY ALL jet_rcpt_qty
      OPERATOR mogas_addition_888
        TRIGGERED BY ALL mogas_rcpt_qty
      OPERATOR mogas_subtraction_891
        TRIGGERED BY ALL mogas_iss_qty
      OPERATOR mogas_gage_894
        TRIGGERED BY SOME mg_qty_on_hand
      OPERATOR other_iss_db_table_504
        TRIGGERED BY SOME eq_iss_fuel_type, eq_iss_qty, eq_iss_id, eq_iss_unit, eq_iss_name
      OPERATOR other_rcpt_db_table_501
        TRIGGERED BY SOME oth_rcpt_fuel_type, oth_rcpt_qty, oth_rcpt_source_id, oth_rcpt_source_unit
      OPERATOR bulk_iss_db_table_498
        TRIGGERED BY SOME bulk_iss_fuel_type, bulk_iss_qty, bulk_iss_doc_num, bulk_rcv_unit,
bulk_rcv_name
      OPERATOR bulk_rcpt_db_table_495
        TRIGGERED BY SOME bulk_rcpt_fuel_type, bulk_rcpt_qty, bulk_rcpt_doc_number
      OPERATOR daily_reporter_410
        PERIOD 3600000 MS
        FINISH WITHIN 750 MS
      OPERATOR df_iss_totalizer_352
        TRIGGERED BY ALL i_df_qty
      OPERATOR mg_iss_totalizer_349
        TRIGGERED BY ALL i_mg_qty
      OPERATOR jet_iss_totalizer_346
        TRIGGERED BY ALL i_jet_qty
      OPERATOR jet_rcpt_totalizer_280
        TRIGGERED BY ALL r_jet_qty
      OPERATOR mg_rcpt_totalizer_277
        TRIGGERED BY ALL r_mg_qty
      OPERATOR df_rcpt_totalizer_274
        TRIGGERED BY ALL r_df_qty
```

```
OPERATOR bulk_rcpt_processor_198
 TRIGGERED BY ALL bulk_rcpt_fuel_type, bulk_rcpt_qty
OPERATOR oth_rcpt_processor_207
 TRIGGERED BY ALL oth_rcpt_fuel_type, oth_rcpt_qty
OPERATOR rcpt_processor_210
 TRIGGERED BY SOME bulk_rcpt_enable, oth_rcpt_enable
OPERATOR oth_iss_processor_307
 TRIGGERED BY ALL eq_iss_qty, eq_iss_fuel_type
OPERATOR bulk_iss_processor_310
 TRIGGERED BY ALL bulk_iss_fuel_type, bulk_iss_qty
OPERATOR iss_processor_323
 TRIGGERED BY SOME bulk_iss_enable, oth_iss_enable
OPERATOR daily_iss_db_table_743
 TRIGGERED BY SOME daily_df_iss_total, daily_mg_iss_total, daily_jet_iss_total
OPERATOR daily_rcpt_db_table_740
 TRIGGERED BY SOME daily_df_rcpt_total, daily_mg_rcpt_total, daily_jet_rcpt_total
OPERATOR monthly_reporter_601
 PERIOD 108000000 MS
 FINISH WITHIN 750 MS
OPERATOR mo_jet_iss_totalizer_598
 TRIGGERED BY SOME daily_jet_iss_total
OPERATOR mo_mg_iss_totalizer_595
 TRIGGERED BY SOME daily_mg_iss_total
OPERATOR mo_df_iss_totalizer_592
 TRIGGERED BY SOME daily_df_iss_total
OPERATOR mo_jet_rcpt_totalizer_589
 TRIGGERED BY SOME daily_jet_rcpt_total
OPERATOR mo_mg_rcpt_totalizer_586
 TRIGGERED BY SOME daily_mg_rcpt_total
OPERATOR mo_df_rcpt_totalizer_583
 TRIGGERED BY SOME daily_df_rcpt_total
OPERATOR diesel_iss_acct_proc_934
 TRIGGERED BY SOME month_df_iss_total
OPERATOR diesel_rcpt_acct_proc_937
 TRIGGERED BY SOME month_df_rcpt_total
OPERATOR mogas_iss_acct_proc_940
 TRIGGERED BY SOME month_mg_iss_total
OPERATOR mogas_rcpt_acct_proc_943
 TRIGGERED BY SOME month_mg_rcpt_total
OPERATOR jet_iss_acct_proc_946
 TRIGGERED BY SOME month_jet_iss_total
OPERATOR jet_rcpt_acct_proc_949
 TRIGGERED BY SOME month_jet_rcpt_total
OPERATOR df_acct_calc_952
 TRIGGERED BY ALL total_mo_df_iss, total_mo_df_rcpt
OPERATOR mg_acct_calc_955
 TRIGGERED BY ALL total_mo_mg_iss, total_mo_mg_rcpt
OPERATOR jet_acct_calc_958
 TRIGGERED BY ALL total_mo_jet_iss, total_mo_jet_rcpt
OPERATOR diesel_gage_854
 TRIGGERED BY SOME df_qty_on_hand
OPERATOR diesel_subtraction_839
 TRIGGERED BY ALL diesel_iss_qty
OPERATOR diesel_addition_836
 TRIGGERED BY ALL diesel_rcpt_qty
```

DESCRIPTION {The Fuel Automated Subsystem automates Army petroleum
management/accountability in accordance with Army Regulation 710-2 and 735-5.}

END

APPENDIX F

```
--------------------------------------------------------------------------------
--
-- File: fuel_subsystem.a
-- Author: Lawrence A. Kominiak, Major, USA
-- Project: Fuel Automated Subsystem of ICS3
-- Date: February 1998
-- Description: Control code generated by CAPS
--
--------------------------------------------------------------------------------

 package FUEL_SUBSYSTEM_1_EXCEPTIONS is
  -- PSDL exception type declaration
  type PSDL_EXCEPTION is (UNDECLARED_ADA_EXCEPTION);
 end FUEL_SUBSYSTEM_1_EXCEPTIONS;

 package FUEL_SUBSYSTEM_1_INSTANTIATIONS is
  -- Ada Generic package instantiations

 end FUEL_SUBSYSTEM_1_INSTANTIATIONS;

  with PSDL_TIMERS;
 package FUEL_SUBSYSTEM_1_TIMERS is
  -- Timer instantiations
 end FUEL_SUBSYSTEM_1_TIMERS;

-- with/use clauses for atomic type packages
  with TEXT_STRING_PKG; use TEXT_STRING_PKG;
-- with/use clauses for generated packages.
  with FUEL_SUBSYSTEM_1_EXCEPTIONS; use FUEL_SUBSYSTEM_1_EXCEPTIONS;
  with FUEL_SUBSYSTEM_1_INSTANTIATIONS; use FUEL_SUBSYSTEM_1_INSTANTIATIONS;
-- with/use clauses for CAPS library packages.
  with PSDL_STREAMS; use PSDL_STREAMS;
 package FUEL_SUBSYSTEM_1_STREAMS is
-- Local stream instantiations

  package DS_JET_QTY_AVAILABLE_JET_ACCT_CALC_958 is new
   PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

  package DS_JET_QTY_AVAILABLE_GUI_FUEL_ON_HAND_124 is new
   PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

  package DS_MOGAS_QTY_AVAILABLE_MG_ACCT_CALC_955 is new
   PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

  package DS_MOGAS_QTY_AVAILABLE_GUI_FUEL_ON_HAND_124 is new
   PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

  package DS_JET_ISS_QTY_JET_SUBTRACTION_914 is new
   PSDL_STREAMS.FIFO_BUFFER(INTEGER);

  package DS_JET_RCPT_QTY_JET_ADDITION_911 is new
```

```
 PSDL_STREAMS.FIFO_BUFFER(INTEGER);

package DS_MOGAS_RCPT_QTY_MOGAS_ADDITION_888 is new
 PSDL_STREAMS.FIFO_BUFFER(INTEGER);

package DS_MOGAS_ISS_QTY_MOGAS_SUBTRACTION_891 is new
 PSDL_STREAMS.FIFO_BUFFER(INTEGER);

package DS_DIESEL_QTY_AVAILABLE_DF_ACCT_CALC_952 is new
 PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_DIESEL_QTY_AVAILABLE_GUI_FUEL_ON_HAND_124 is new
 PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_MONTH_DF_ISS_TOTAL_DIESEL_ISS_ACCT_PROC_934 is new
 PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_MONTH_DF_RCPT_TOTAL_DIESEL_RCPT_ACCT_PROC_937 is new
 PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_DAILY_DF_RCPT_TOTAL_MO_DF_RCPT_TOTALIZER_583 is new
 PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_DAILY_DF_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740 is new
 PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_DAILY_DF_ISS_TOTAL_MO_DF_ISS_TOTALIZER_592 is new
 PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_DAILY_DF_ISS_TOTAL_DAILY_ISS_DB_TABLE_743 is new
 PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_EQ_ISS_UNIT_OTHER_ISS_DB_TABLE_504 is new
 PSDL_STREAMS.SAMPLED_BUFFER(TEXT_STRING);

package DS_EQ_ISS_NAME_OTHER_ISS_DB_TABLE_504 is new
 PSDL_STREAMS.SAMPLED_BUFFER(TEXT_STRING);

package DS_EQ_ISS_FUEL_TYPE_ISS_PROCESSOR_323 is new
 PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_EQ_ISS_FUEL_TYPE_OTH_ISS_PROCESSOR_307 is new
 PSDL_STREAMS.FIFO_BUFFER(INTEGER);

package DS_EQ_ISS_FUEL_TYPE_OTHER_ISS_DB_TABLE_504 is new
 PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_EQ_ISS_ID_OTHER_ISS_DB_TABLE_504 is new
 PSDL_STREAMS.SAMPLED_BUFFER(TEXT_STRING);

package DS_EQ_ISS_QTY_ISS_PROCESSOR_323 is new
 PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_EQ_ISS_QTY_OTH_ISS_PROCESSOR_307 is new
 PSDL_STREAMS.FIFO_BUFFER(INTEGER);
```

```
package DS_EQ_ISS_QTY_OTHER_ISS_DB_TABLE_504 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_BULK_RCV_UNIT_BULK_ISS_DB_TABLE_498 is new
  PSDL_STREAMS.SAMPLED_BUFFER(TEXT_STRING);

package DS_BULK_ISS_DOC_NUM_BULK_ISS_DB_TABLE_498 is new
  PSDL_STREAMS.SAMPLED_BUFFER(TEXT_STRING);

package DS_BULK_RCV_NAME_BULK_ISS_DB_TABLE_498 is new
  PSDL_STREAMS.SAMPLED_BUFFER(TEXT_STRING);

package DS_BULK_ISS_QTY_ISS_PROCESSOR_323 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_BULK_ISS_QTY_BULK_ISS_PROCESSOR_310 is new
  PSDL_STREAMS.FIFO_BUFFER(INTEGER);

package DS_BULK_ISS_QTY_BULK_ISS_DB_TABLE_498 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_BULK_ISS_FUEL_TYPE_ISS_PROCESSOR_323 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_BULK_ISS_FUEL_TYPE_BULK_ISS_PROCESSOR_310 is new
  PSDL_STREAMS.FIFO_BUFFER(INTEGER);

package DS_BULK_ISS_FUEL_TYPE_BULK_ISS_DB_TABLE_498 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_OTH_RCPT_QTY_RCPT_PROCESSOR_210 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_OTH_RCPT_QTY_OTH_RCPT_PROCESSOR_207 is new
  PSDL_STREAMS.FIFO_BUFFER(INTEGER);

package DS_OTH_RCPT_QTY_OTHER_RCPT_DB_TABLE_501 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_OTH_RCPT_SOURCE_UNIT_OTHER_RCPT_DB_TABLE_501 is new
  PSDL_STREAMS.SAMPLED_BUFFER(TEXT_STRING);

package DS_OTH_RCPT_FUEL_TYPE_RCPT_PROCESSOR_210 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_OTH_RCPT_FUEL_TYPE_OTH_RCPT_PROCESSOR_207 is new
  PSDL_STREAMS.FIFO_BUFFER(INTEGER);

package DS_OTH_RCPT_FUEL_TYPE_OTHER_RCPT_DB_TABLE_501 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_OTH_RCPT_SOURCE_ID_OTHER_RCPT_DB_TABLE_501 is new
  PSDL_STREAMS.SAMPLED_BUFFER(TEXT_STRING);

package DS_BULK_RCPT_FUEL_TYPE_RCPT_PROCESSOR_210 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);
```

```
package DS_BULK_RCPT_FUEL_TYPE_BULK_RCPT_PROCESSOR_198 is new
  PSDL_STREAMS.FIFO_BUFFER(INTEGER);

package DS_BULK_RCPT_FUEL_TYPE_BULK_RCPT_DB_TABLE_495 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_BULK_RCPT_DOC_NUMBER_BULK_RCPT_DB_TABLE_495 is new
  PSDL_STREAMS.SAMPLED_BUFFER(TEXT_STRING);

package DS_BULK_RCPT_QTY_RCPT_PROCESSOR_210 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_BULK_RCPT_QTY_BULK_RCPT_PROCESSOR_198 is new
  PSDL_STREAMS.FIFO_BUFFER(INTEGER);

package DS_BULK_RCPT_QTY_BULK_RCPT_DB_TABLE_495 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_DAILY_JET_ISS_TOTAL_MO_JET_ISS_TOTALIZER_598 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_DAILY_JET_ISS_TOTAL_DAILY_ISS_DB_TABLE_743 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_DAILY_JET_RCPT_TOTAL_MO_JET_RCPT_TOTALIZER_589 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_DAILY_JET_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_DAILY_MG_ISS_TOTAL_MO_MG_ISS_TOTALIZER_595 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_DAILY_MG_ISS_TOTAL_DAILY_ISS_DB_TABLE_743 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_DAILY_MG_RCPT_TOTAL_MO_MG_RCPT_TOTALIZER_586 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_DAILY_MG_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_MONTH_JET_ISS_TOTAL_JET_ISS_ACCT_PROC_946 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_MONTH_JET_RCPT_TOTAL_JET_RCPT_ACCT_PROC_949 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_MONTH_MG_ISS_TOTAL_MOGAS_ISS_ACCT_PROC_940 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_MONTH_MG_RCPT_TOTAL_MOGAS_RCPT_ACCT_PROC_943 is new
  PSDL_STREAMS.SAMPLED_BUFFER(INTEGER);

package DS_TOLERANCE_DF_GUI_ACC_OFFICER_179 is new
```

```
  PSDL_STREAMS.SAMPLED_BUFFER(BOOLEAN);

package DS_TOLERANCE_MG_GUI_ACC_OFFICER_179 is new
  PSDL_STREAMS.SAMPLED_BUFFER(BOOLEAN);

package DS_TOLERANCE_JET_GUI_ACC_OFFICER_179 is new
  PSDL_STREAMS.SAMPLED_BUFFER(BOOLEAN);

package DS_DIESEL_RCPT_QTY_DIESEL_ADDITION_836 is new
  PSDL_STREAMS.FIFO_BUFFER(INTEGER);

package DS_DIESEL_ISS_QTY_DIESEL_SUBTRACTION_839 is new
  PSDL_STREAMS.FIFO_BUFFER(INTEGER);

package DS_I_DF_QTY_DF_ISS_TOTALIZER_352 is new
  PSDL_STREAMS.FIFO_BUFFER(INTEGER);

package DS_I_MG_QTY_MG_ISS_TOTALIZER_349 is new
  PSDL_STREAMS.FIFO_BUFFER(INTEGER);

package DS_I_JET_QTY_JET_ISS_TOTALIZER_346 is new
  PSDL_STREAMS.FIFO_BUFFER(INTEGER);

package DS_R_JET_QTY_JET_RCPT_TOTALIZER_280 is new
  PSDL_STREAMS.FIFO_BUFFER(INTEGER);

package DS_R_MG_QTY_MG_RCPT_TOTALIZER_277 is new
  PSDL_STREAMS.FIFO_BUFFER(INTEGER);

package DS_R_DF_QTY_DF_RCPT_TOTALIZER_274 is new
  PSDL_STREAMS.FIFO_BUFFER(INTEGER);

-- State stream instantiations

package DS_JET_QTY_ON_HAND_JET_GAGE_917 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_JET_VOLUME_JET_SUBTRACTION_914 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_JET_VOLUME_JET_ADDITION_911 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MOGAS_VOLUME_MOGAS_SUBTRACTION_891 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MOGAS_VOLUME_MOGAS_ADDITION_888 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MG_QTY_ON_HAND_MOGAS_GAGE_894 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_JET_ISS_TOTAL_JET_ISS_TOTALIZER_346 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_JET_ISS_TOTAL_DAILY_REPORTER_410 is new
```

```
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MG_ISS_TOTAL_MG_ISS_TOTALIZER_349 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MG_ISS_TOTAL_DAILY_REPORTER_410 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_DF_ISS_TOTAL_DF_ISS_TOTALIZER_352 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_DF_ISS_TOTAL_DAILY_REPORTER_410 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_JET_RCPT_TOTAL_JET_RCPT_TOTALIZER_280 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_JET_RCPT_TOTAL_DAILY_REPORTER_410 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MG_RCPT_TOTAL_MG_RCPT_TOTALIZER_277 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MG_RCPT_TOTAL_DAILY_REPORTER_410 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_DF_RCPT_TOTAL_DF_RCPT_TOTALIZER_274 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_DF_RCPT_TOTAL_DAILY_REPORTER_410 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_OTH_RCPT_ENABLE_BULK_RCPT_PROCESSOR_198 is new
  PSDL_STREAMS.STATE_VARIABLE(BOOLEAN, false);

package DS_OTH_RCPT_ENABLE_RCPT_PROCESSOR_210 is new
  PSDL_STREAMS.STATE_VARIABLE(BOOLEAN, false);

package DS_BULK_RCPT_ENABLE_OTH_RCPT_PROCESSOR_207 is new
  PSDL_STREAMS.STATE_VARIABLE(BOOLEAN, false);

package DS_BULK_RCPT_ENABLE_RCPT_PROCESSOR_210 is new
  PSDL_STREAMS.STATE_VARIABLE(BOOLEAN, false);

package DS_BULK_ISS_ENABLE_OTH_ISS_PROCESSOR_307 is new
  PSDL_STREAMS.STATE_VARIABLE(BOOLEAN, false);

package DS_BULK_ISS_ENABLE_ISS_PROCESSOR_323 is new
  PSDL_STREAMS.STATE_VARIABLE(BOOLEAN, false);

package DS_OTH_ISS_ENABLE_BULK_ISS_PROCESSOR_310 is new
  PSDL_STREAMS.STATE_VARIABLE(BOOLEAN, false);

package DS_OTH_ISS_ENABLE_ISS_PROCESSOR_323 is new
  PSDL_STREAMS.STATE_VARIABLE(BOOLEAN, false);
```

```
package DS_MO_ISS_JET_TOTAL_MO_JET_ISS_TOTALIZER_598 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MO_ISS_JET_TOTAL_MONTHLY_REPORTER_601 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MO_ISS_MG_TOTAL_MO_MG_ISS_TOTALIZER_595 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MO_ISS_MG_TOTAL_MONTHLY_REPORTER_601 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MO_ISS_DF_TOTAL_MO_DF_ISS_TOTALIZER_592 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MO_ISS_DF_TOTAL_MONTHLY_REPORTER_601 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MO_RCPT_JET_TOTAL_MO_JET_RCPT_TOTALIZER_589 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MO_RCPT_JET_TOTAL_MONTHLY_REPORTER_601 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MO_RCPT_MG_TOTAL_MO_MG_RCPT_TOTALIZER_586 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MO_RCPT_MG_TOTAL_MONTHLY_REPORTER_601 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MO_RCPT_DF_TOTAL_MO_DF_RCPT_TOTALIZER_583 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_MO_RCPT_DF_TOTAL_MONTHLY_REPORTER_601 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_TOTAL_MO_DF_ISS_DF_ACCT_CALC_952 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_TOTAL_MO_DF_RCPT_DF_ACCT_CALC_952 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_TOTAL_MO_MG_ISS_MG_ACCT_CALC_955 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_TOTAL_MO_MG_RCPT_MG_ACCT_CALC_955 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_TOTAL_MO_JET_ISS_JET_ACCT_CALC_958 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_TOTAL_MO_JET_RCPT_JET_ACCT_CALC_958 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

package DS_OPENING_INV_DIESEL_DF_ACCT_CALC_952 is new
  PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);
```

```
 package DS_OPENING_INV_MOGAS_MG_ACCT_CALC_955 is new
   PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

 package DS_OPENING_INV_JET_JET_ACCT_CALC_958 is new
   PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

 package DS_DF_QTY_ON_HAND_DIESEL_GAGE_854 is new
   PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

 package DS_DIESEL_VOLUME_DIESEL_ADDITION_836 is new
   PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

 package DS_DIESEL_VOLUME_DIESEL_SUBTRACTION_839 is new
   PSDL_STREAMS.STATE_VARIABLE(INTEGER, 0);

end FUEL_SUBSYSTEM_1_STREAMS;

package FUEL_SUBSYSTEM_1_DRIVERS is
 procedure GUI_BULK_RECEIPT_3_DRIVER;
 procedure GUI_OTHER_RECEIPT_6_DRIVER;
 procedure GUI_BULK_ISSUE_9_DRIVER;
 procedure GUI_OTHER_ISSUE_12_DRIVER;
 procedure GUI_FUEL_ON_HAND_124_DRIVER;
 procedure GUI_ACC_OFFICER_179_DRIVER;
 procedure JET_GAGE_917_DRIVER;
 procedure JET_SUBTRACTION_914_DRIVER;
 procedure JET_ADDITION_911_DRIVER;
 procedure MOGAS_ADDITION_888_DRIVER;
 procedure MOGAS_SUBTRACTION_891_DRIVER;
 procedure MOGAS_GAGE_894_DRIVER;
 procedure OTHER_ISS_DB_TABLE_504_DRIVER;
 procedure OTHER_RCPT_DB_TABLE_501_DRIVER;
 procedure BULK_ISS_DB_TABLE_498_DRIVER;
 procedure BULK_RCPT_DB_TABLE_495_DRIVER;
 procedure DAILY_REPORTER_410_DRIVER;
 procedure DF_ISS_TOTALIZER_352_DRIVER;
 procedure MG_ISS_TOTALIZER_349_DRIVER;
 procedure JET_ISS_TOTALIZER_346_DRIVER;
 procedure JET_RCPT_TOTALIZER_280_DRIVER;
 procedure MG_RCPT_TOTALIZER_277_DRIVER;
 procedure DF_RCPT_TOTALIZER_274_DRIVER;
 procedure BULK_RCPT_PROCESSOR_198_DRIVER;
 procedure OTH_RCPT_PROCESSOR_207_DRIVER;
 procedure RCPT_PROCESSOR_210_DRIVER;
 procedure OTH_ISS_PROCESSOR_307_DRIVER;
 procedure BULK_ISS_PROCESSOR_310_DRIVER;
 procedure ISS_PROCESSOR_323_DRIVER;
 procedure DAILY_ISS_DB_TABLE_743_DRIVER;
 procedure DAILY_RCPT_DB_TABLE_740_DRIVER;
 procedure MONTHLY_REPORTER_601_DRIVER;
 procedure MO_JET_ISS_TOTALIZER_598_DRIVER;
 procedure MO_MG_ISS_TOTALIZER_595_DRIVER;
 procedure MO_DF_ISS_TOTALIZER_592_DRIVER;
 procedure MO_JET_RCPT_TOTALIZER_589_DRIVER;
 procedure MO_MG_RCPT_TOTALIZER_586_DRIVER;
```

```
  procedure MO_DF_RCPT_TOTALIZER_583_DRIVER;
  procedure DIESEL_ISS_ACCT_PROC_934_DRIVER;
  procedure DIESEL_RCPT_ACCT_PROC_937_DRIVER;
  procedure MOGAS_ISS_ACCT_PROC_940_DRIVER;
  procedure MOGAS_RCPT_ACCT_PROC_943_DRIVER;
  procedure JET_ISS_ACCT_PROC_946_DRIVER;
  procedure JET_RCPT_ACCT_PROC_949_DRIVER;
  procedure DF_ACCT_CALC_952_DRIVER;
  procedure MG_ACCT_CALC_955_DRIVER;
  procedure JET_ACCT_CALC_958_DRIVER;
  procedure DIESEL_GAGE_854_DRIVER;
  procedure DIESEL_SUBTRACTION_839_DRIVER;
  procedure DIESEL_ADDITION_836_DRIVER;
 end FUEL_SUBSYSTEM_1_DRIVERS;

-- with/use clauses for atomic components.
  with TEXT_STRING_PKG; use TEXT_STRING_PKG;
  with OTHER_ISS_DB_TABLE_504_PKG; use OTHER_ISS_DB_TABLE_504_PKG;
  with OTHER_RCPT_DB_TABLE_501_PKG; use OTHER_RCPT_DB_TABLE_501_PKG;
  with BULK_ISS_DB_TABLE_498_PKG; use BULK_ISS_DB_TABLE_498_PKG;
  with BULK_RCPT_DB_TABLE_495_PKG; use BULK_RCPT_DB_TABLE_495_PKG;
  with DAILY_REPORTER_410_PKG; use DAILY_REPORTER_410_PKG;
  with DF_ISS_TOTALIZER_352_PKG; use DF_ISS_TOTALIZER_352_PKG;
  with MG_ISS_TOTALIZER_349_PKG; use MG_ISS_TOTALIZER_349_PKG;
  with JET_ISS_TOTALIZER_346_PKG; use JET_ISS_TOTALIZER_346_PKG;
  with JET_RCPT_TOTALIZER_280_PKG; use JET_RCPT_TOTALIZER_280_PKG;
  with MG_RCPT_TOTALIZER_277_PKG; use MG_RCPT_TOTALIZER_277_PKG;
  with DF_RCPT_TOTALIZER_274_PKG; use DF_RCPT_TOTALIZER_274_PKG;
  with BULK_RCPT_PROCESSOR_198_PKG; use BULK_RCPT_PROCESSOR_198_PKG;
  with OTH_RCPT_PROCESSOR_207_PKG; use OTH_RCPT_PROCESSOR_207_PKG;
  with RCPT_PROCESSOR_210_PKG; use RCPT_PROCESSOR_210_PKG;
  with OTH_ISS_PROCESSOR_307_PKG; use OTH_ISS_PROCESSOR_307_PKG;
  with BULK_ISS_PROCESSOR_310_PKG; use BULK_ISS_PROCESSOR_310_PKG;
  with ISS_PROCESSOR_323_PKG; use ISS_PROCESSOR_323_PKG;
  with JET_GAGE_917_PKG; use JET_GAGE_917_PKG;
  with JET_SUBTRACTION_914_PKG; use JET_SUBTRACTION_914_PKG;
  with JET_ADDITION_911_PKG; use JET_ADDITION_911_PKG;
  with MOGAS_ADDITION_888_PKG; use MOGAS_ADDITION_888_PKG;
  with MOGAS_SUBTRACTION_891_PKG; use MOGAS_SUBTRACTION_891_PKG;
  with MOGAS_GAGE_894_PKG; use MOGAS_GAGE_894_PKG;
  with GUI_BULK_RECEIPT_3_PKG; use GUI_BULK_RECEIPT_3_PKG;
  with GUI_OTHER_RECEIPT_6_PKG; use GUI_OTHER_RECEIPT_6_PKG;
  with GUI_BULK_ISSUE_9_PKG; use GUI_BULK_ISSUE_9_PKG;
  with GUI_OTHER_ISSUE_12_PKG; use GUI_OTHER_ISSUE_12_PKG;
  with DAILY_ISS_DB_TABLE_743_PKG; use DAILY_ISS_DB_TABLE_743_PKG;
  with DAILY_RCPT_DB_TABLE_740_PKG; use DAILY_RCPT_DB_TABLE_740_PKG;
  with MONTHLY_REPORTER_601_PKG; use MONTHLY_REPORTER_601_PKG;
  with MO_JET_ISS_TOTALIZER_598_PKG; use MO_JET_ISS_TOTALIZER_598_PKG;
  with MO_MG_ISS_TOTALIZER_595_PKG; use MO_MG_ISS_TOTALIZER_595_PKG;
  with MO_DF_ISS_TOTALIZER_592_PKG; use MO_DF_ISS_TOTALIZER_592_PKG;
  with MO_JET_RCPT_TOTALIZER_589_PKG; use MO_JET_RCPT_TOTALIZER_589_PKG;
  with MO_MG_RCPT_TOTALIZER_586_PKG; use MO_MG_RCPT_TOTALIZER_586_PKG;
  with MO_DF_RCPT_TOTALIZER_583_PKG; use MO_DF_RCPT_TOTALIZER_583_PKG;
  with DIESEL_ISS_ACCT_PROC_934_PKG; use DIESEL_ISS_ACCT_PROC_934_PKG;
  with DIESEL_RCPT_ACCT_PROC_937_PKG; use DIESEL_RCPT_ACCT_PROC_937_PKG;
  with MOGAS_ISS_ACCT_PROC_940_PKG; use MOGAS_ISS_ACCT_PROC_940_PKG;
```

```
  with MOGAS_RCPT_ACCT_PROC_943_PKG; use MOGAS_RCPT_ACCT_PROC_943_PKG;
  with JET_ISS_ACCT_PROC_946_PKG; use JET_ISS_ACCT_PROC_946_PKG;
  with JET_RCPT_ACCT_PROC_949_PKG; use JET_RCPT_ACCT_PROC_949_PKG;
  with DF_ACCT_CALC_952_PKG; use DF_ACCT_CALC_952_PKG;
  with MG_ACCT_CALC_955_PKG; use MG_ACCT_CALC_955_PKG;
  with JET_ACCT_CALC_958_PKG; use JET_ACCT_CALC_958_PKG;
  with GUI_FUEL_ON_HAND_124_PKG; use GUI_FUEL_ON_HAND_124_PKG;
  with GUI_ACC_OFFICER_179_PKG; use GUI_ACC_OFFICER_179_PKG;
  with DIESEL_GAGE_854_PKG; use DIESEL_GAGE_854_PKG;
  with DIESEL_SUBTRACTION_839_PKG; use DIESEL_SUBTRACTION_839_PKG;
  with DIESEL_ADDITION_836_PKG; use DIESEL_ADDITION_836_PKG;
-- with/use clauses for generated packages.
  with FUEL_SUBSYSTEM_1_EXCEPTIONS; use FUEL_SUBSYSTEM_1_EXCEPTIONS;
  with FUEL_SUBSYSTEM_1_STREAMS; use FUEL_SUBSYSTEM_1_STREAMS;
  with FUEL_SUBSYSTEM_1_TIMERS; use FUEL_SUBSYSTEM_1_TIMERS;
  with FUEL_SUBSYSTEM_1_INSTANTIATIONS; use FUEL_SUBSYSTEM_1_INSTANTIATIONS;
-- with/use clauses for CAPS library packages.
  with DS_DEBUG_PKG; use DS_DEBUG_PKG;
  with PSDL_STREAMS; use PSDL_STREAMS;
  with PSDL_TIMERS;
 package body FUEL_SUBSYSTEM_1_DRIVERS is

  procedure GUI_BULK_RECEIPT_3_DRIVER is
   LV_BULK_RCPT_FUEL_TYPE : INTEGER;
   LV_BULK_RCPT_DOC_NUMBER : TEXT_STRING_PKG.TEXT_STRING;
   LV_BULK_RCPT_QTY : INTEGER;

   EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
   EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.

-- Data stream reads.

-- Execution trigger condition check.
    if True then
     begin
     GUI_BULK_RECEIPT_3(
      BULK_RCPT_FUEL_TYPE => LV_BULK_RCPT_FUEL_TYPE,
      BULK_RCPT_DOC_NUMBER => LV_BULK_RCPT_DOC_NUMBER,
      BULK_RCPT_QTY => LV_BULK_RCPT_QTY);
     exception
      when others =>
       DS_DEBUG.UNDECLARED_EXCEPTION("GUI_BULK_RECEIPT_3");
       EXCEPTION_HAS_OCCURRED := true;
       EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
     end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
```

```
      begin

DS_BULK_RCPT_FUEL_TYPE_RCPT_PROCESSOR_210.BUFFER.WRITE(LV_BULK_RCPT_FUEL
_TYPE);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("BULK_RCPT_FUEL_TYPE_RCPT_PROCESSOR_210",
"GUI_BULK_RECEIPT_3");
      end;
      begin

DS_BULK_RCPT_FUEL_TYPE_BULK_RCPT_PROCESSOR_198.BUFFER.WRITE(LV_BULK_RCP
T_FUEL_TYPE);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("BULK_RCPT_FUEL_TYPE_BULK_RCPT_PROCESSOR_198",
"GUI_BULK_RECEIPT_3");
      end;
      begin

DS_BULK_RCPT_FUEL_TYPE_BULK_RCPT_DB_TABLE_495.BUFFER.WRITE(LV_BULK_RCPT_
FUEL_TYPE);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("BULK_RCPT_FUEL_TYPE_BULK_RCPT_DB_TABLE_495",
"GUI_BULK_RECEIPT_3");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_BULK_RCPT_DOC_NUMBER_BULK_RCPT_DB_TABLE_495.BUFFER.WRITE(LV_BULK_RC
PT_DOC_NUMBER);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("BULK_RCPT_DOC_NUMBER_BULK_RCPT_DB_TABLE_495
", "GUI_BULK_RECEIPT_3");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_BULK_RCPT_QTY_RCPT_PROCESSOR_210.BUFFER.WRITE(LV_BULK_RCPT_QTY);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("BULK_RCPT_QTY_RCPT_PROCESSOR_210",
"GUI_BULK_RECEIPT_3");
      end;
      begin

DS_BULK_RCPT_QTY_BULK_RCPT_PROCESSOR_198.BUFFER.WRITE(LV_BULK_RCPT_QTY);
      exception
        when BUFFER_OVERFLOW =>
```

```
        DS_DEBUG.BUFFER_OVERFLOW("BULK_RCPT_QTY_BULK_RCPT_PROCESSOR_198",
"GUI_BULK_RECEIPT_3");
      end;
      begin

DS_BULK_RCPT_QTY_BULK_RCPT_DB_TABLE_495.BUFFER.WRITE(LV_BULK_RCPT_QTY);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("BULK_RCPT_QTY_BULK_RCPT_DB_TABLE_495",
"GUI_BULK_RECEIPT_3");
      end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "GUI_BULK_RECEIPT_3",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end GUI_BULK_RECEIPT_3_DRIVER;


  procedure GUI_OTHER_RECEIPT_6_DRIVER is
    LV_OTH_RCPT_QTY : INTEGER;
    LV_OTH_RCPT_SOURCE_UNIT : TEXT_STRING_PKG.TEXT_STRING;
    LV_OTH_RCPT_FUEL_TYPE : INTEGER;
    LV_OTH_RCPT_SOURCE_ID : TEXT_STRING_PKG.TEXT_STRING;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.

-- Data stream reads.

-- Execution trigger condition check.
    if True then
      begin
      GUI_OTHER_RECEIPT_6(
        OTH_RCPT_QTY => LV_OTH_RCPT_QTY,
        OTH_RCPT_SOURCE_UNIT => LV_OTH_RCPT_SOURCE_UNIT,
        OTH_RCPT_FUEL_TYPE => LV_OTH_RCPT_FUEL_TYPE,
        OTH_RCPT_SOURCE_ID => LV_OTH_RCPT_SOURCE_ID);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("GUI_OTHER_RECEIPT_6");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.
```

```
--Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_OTH_RCPT_QTY_RCPT_PROCESSOR_210.BUFFER.WRITE(LV_OTH_RCPT_QTY);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("OTH_RCPT_QTY_RCPT_PROCESSOR_210",
"GUI_OTHER_RECEIPT_6");
      end;
      begin

DS_OTH_RCPT_QTY_OTH_RCPT_PROCESSOR_207.BUFFER.WRITE(LV_OTH_RCPT_QTY);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("OTH_RCPT_QTY_OTH_RCPT_PROCESSOR_207",
"GUI_OTHER_RECEIPT_6");
      end;
      begin

DS_OTH_RCPT_QTY_OTHER_RCPT_DB_TABLE_501.BUFFER.WRITE(LV_OTH_RCPT_QTY);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("OTH_RCPT_QTY_OTHER_RCPT_DB_TABLE_501",
"GUI_OTHER_RECEIPT_6");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_OTH_RCPT_SOURCE_UNIT_OTHER_RCPT_DB_TABLE_501.BUFFER.WRITE(LV_OTH_RCP
T_SOURCE_UNIT);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("OTH_RCPT_SOURCE_UNIT_OTHER_RCPT_DB_TABLE_501
", "GUI_OTHER_RECEIPT_6");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_OTH_RCPT_FUEL_TYPE_RCPT_PROCESSOR_210.BUFFER.WRITE(LV_OTH_RCPT_FUEL_T
YPE);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("OTH_RCPT_FUEL_TYPE_RCPT_PROCESSOR_210",
"GUI_OTHER_RECEIPT_6");
      end;
      begin

DS_OTH_RCPT_FUEL_TYPE_OTH_RCPT_PROCESSOR_207.BUFFER.WRITE(LV_OTH_RCPT_FU
EL_TYPE);
      exception
        when BUFFER_OVERFLOW =>
```

```
DS_DEBUG.BUFFER_OVERFLOW("OTH_RCPT_FUEL_TYPE_OTH_RCPT_PROCESSOR_207",
"GUI_OTHER_RECEIPT_6");
       end;
       begin

DS_OTH_RCPT_FUEL_TYPE_OTHER_RCPT_DB_TABLE_501.BUFFER.WRITE(LV_OTH_RCPT_F
UEL_TYPE);
       exception
         when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("OTH_RCPT_FUEL_TYPE_OTHER_RCPT_DB_TABLE_501",
"GUI_OTHER_RECEIPT_6");
       end;
     end if;
     if not EXCEPTION_HAS_OCCURRED then
       begin

DS_OTH_RCPT_SOURCE_ID_OTHER_RCPT_DB_TABLE_501.BUFFER.WRITE(LV_OTH_RCPT_S
OURCE_ID);
       exception
         when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("OTH_RCPT_SOURCE_ID_OTHER_RCPT_DB_TABLE_501",
"GUI_OTHER_RECEIPT_6");
       end;
     end if;

 -- PSDL Exception handler.
     if EXCEPTION_HAS_OCCURRED then
       DS_DEBUG.UNHANDLED_EXCEPTION(
         "GUI_OTHER_RECEIPT_6",
         PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
     end if;
   end GUI_OTHER_RECEIPT_6_DRIVER;


   procedure GUI_BULK_ISSUE_9_DRIVER is
     LV_BULK_RCV_UNIT : TEXT_STRING_PKG.TEXT_STRING;
     LV_BULK_ISS_DOC_NUM : TEXT_STRING_PKG.TEXT_STRING;
     LV_BULK_RCV_NAME : TEXT_STRING_PKG.TEXT_STRING;
     LV_BULK_ISS_QTY : INTEGER;
     LV_BULK_ISS_FUEL_TYPE : INTEGER;

     EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
     EXCEPTION_ID: PSDL_EXCEPTION;
   begin
 -- Data trigger checks.

 -- Data stream reads.

 -- Execution trigger condition check.
     if True then
       begin
       GUI_BULK_ISSUE_9(
         BULK_RCV_UNIT => LV_BULK_RCV_UNIT,
```

```
      BULK_ISS_DOC_NUM => LV_BULK_ISS_DOC_NUM,
      BULK_RCV_NAME => LV_BULK_RCV_NAME,
      BULK_ISS_QTY => LV_BULK_ISS_QTY,
      BULK_ISS_FUEL_TYPE => LV_BULK_ISS_FUEL_TYPE);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("GUI_BULK_ISSUE_9");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin

DS_BULK_RCV_UNIT_BULK_ISS_DB_TABLE_498.BUFFER.WRITE(LV_BULK_RCV_UNIT);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("BULK_RCV_UNIT_BULK_ISS_DB_TABLE_498",
"GUI_BULK_ISSUE_9");
    end;
  end if;
  if not EXCEPTION_HAS_OCCURRED then
    begin

DS_BULK_ISS_DOC_NUM_BULK_ISS_DB_TABLE_498.BUFFER.WRITE(LV_BULK_ISS_DOC_N
UM);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("BULK_ISS_DOC_NUM_BULK_ISS_DB_TABLE_498",
"GUI_BULK_ISSUE_9");
    end;
  end if;
  if not EXCEPTION_HAS_OCCURRED then
    begin

DS_BULK_RCV_NAME_BULK_ISS_DB_TABLE_498.BUFFER.WRITE(LV_BULK_RCV_NAME);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("BULK_RCV_NAME_BULK_ISS_DB_TABLE_498",
"GUI_BULK_ISSUE_9");
    end;
  end if;
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_BULK_ISS_QTY_ISS_PROCESSOR_323.BUFFER.WRITE(LV_BULK_ISS_QTY);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("BULK_ISS_QTY_ISS_PROCESSOR_323",
"GUI_BULK_ISSUE_9");
```

```
      end;
      begin
        DS_BULK_ISS_QTY_BULK_ISS_PROCESSOR_310.BUFFER.WRITE(LV_BULK_ISS_QTY);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("BULK_ISS_QTY_BULK_ISS_PROCESSOR_310",
"GUI_BULK_ISSUE_9");
      end;
      begin
        DS_BULK_ISS_QTY_BULK_ISS_DB_TABLE_498.BUFFER.WRITE(LV_BULK_ISS_QTY);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("BULK_ISS_QTY_BULK_ISS_DB_TABLE_498",
"GUI_BULK_ISSUE_9");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_BULK_ISS_FUEL_TYPE_ISS_PROCESSOR_323.BUFFER.WRITE(LV_BULK_ISS_FUEL_TYPE
);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("BULK_ISS_FUEL_TYPE_ISS_PROCESSOR_323",
"GUI_BULK_ISSUE_9");
      end;
      begin

DS_BULK_ISS_FUEL_TYPE_BULK_ISS_PROCESSOR_310.BUFFER.WRITE(LV_BULK_ISS_FUEL
_TYPE);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("BULK_ISS_FUEL_TYPE_BULK_ISS_PROCESSOR_310",
"GUI_BULK_ISSUE_9");
      end;
      begin

DS_BULK_ISS_FUEL_TYPE_BULK_ISS_DB_TABLE_498.BUFFER.WRITE(LV_BULK_ISS_FUEL_
TYPE);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("BULK_ISS_FUEL_TYPE_BULK_ISS_DB_TABLE_498",
"GUI_BULK_ISSUE_9");
      end;
    end if;

 -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "GUI_BULK_ISSUE_9",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end GUI_BULK_ISSUE_9_DRIVER;
```

```
procedure GUI_OTHER_ISSUE_12_DRIVER is
  LV_EQ_ISS_UNIT : TEXT_STRING_PKG.TEXT_STRING;
  LV_EQ_ISS_NAME : TEXT_STRING_PKG.TEXT_STRING;
  LV_EQ_ISS_FUEL_TYPE : INTEGER;
  LV_EQ_ISS_ID : TEXT_STRING_PKG.TEXT_STRING;
  LV_EQ_ISS_QTY : INTEGER;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.

-- Execution trigger condition check.
    if True then
      begin
      GUI_OTHER_ISSUE_12(
        EQ_ISS_UNIT => LV_EQ_ISS_UNIT,
        EQ_ISS_NAME => LV_EQ_ISS_NAME,
        EQ_ISS_FUEL_TYPE => LV_EQ_ISS_FUEL_TYPE,
        EQ_ISS_ID => LV_EQ_ISS_ID,
        EQ_ISS_QTY => LV_EQ_ISS_QTY);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("GUI_OTHER_ISSUE_12");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_EQ_ISS_UNIT_OTHER_ISS_DB_TABLE_504.BUFFER.WRITE(LV_EQ_ISS_UNIT);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("EQ_ISS_UNIT_OTHER_ISS_DB_TABLE_504",
"GUI_OTHER_ISSUE_12");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_EQ_ISS_NAME_OTHER_ISS_DB_TABLE_504.BUFFER.WRITE(LV_EQ_ISS_NAME);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("EQ_ISS_NAME_OTHER_ISS_DB_TABLE_504",
"GUI_OTHER_ISSUE_12");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
```

```
      begin
DS_EQ_ISS_FUEL_TYPE_ISS_PROCESSOR_323.BUFFER.WRITE(LV_EQ_ISS_FUEL_TYPE);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("EQ_ISS_FUEL_TYPE_ISS_PROCESSOR_323",
"GUI_OTHER_ISSUE_12");
      end;
      begin
DS_EQ_ISS_FUEL_TYPE_OTH_ISS_PROCESSOR_307.BUFFER.WRITE(LV_EQ_ISS_FUEL_TYPE)
;
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("EQ_ISS_FUEL_TYPE_OTH_ISS_PROCESSOR_307",
"GUI_OTHER_ISSUE_12");
      end;
      begin
DS_EQ_ISS_FUEL_TYPE_OTHER_ISS_DB_TABLE_504.BUFFER.WRITE(LV_EQ_ISS_FUEL_TYP
E);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("EQ_ISS_FUEL_TYPE_OTHER_ISS_DB_TABLE_504",
"GUI_OTHER_ISSUE_12");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_EQ_ISS_ID_OTHER_ISS_DB_TABLE_504.BUFFER.WRITE(LV_EQ_ISS_ID);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("EQ_ISS_ID_OTHER_ISS_DB_TABLE_504",
"GUI_OTHER_ISSUE_12");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_EQ_ISS_QTY_ISS_PROCESSOR_323.BUFFER.WRITE(LV_EQ_ISS_QTY);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("EQ_ISS_QTY_ISS_PROCESSOR_323",
"GUI_OTHER_ISSUE_12");
      end;
      begin
        DS_EQ_ISS_QTY_OTH_ISS_PROCESSOR_307.BUFFER.WRITE(LV_EQ_ISS_QTY);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("EQ_ISS_QTY_OTH_ISS_PROCESSOR_307",
"GUI_OTHER_ISSUE_12");
      end;
      begin
        DS_EQ_ISS_QTY_OTHER_ISS_DB_TABLE_504.BUFFER.WRITE(LV_EQ_ISS_QTY);
      exception
        when BUFFER_OVERFLOW =>
```

```
        DS_DEBUG.BUFFER_OVERFLOW("EQ_ISS_QTY_OTHER_ISS_DB_TABLE_504",
"GUI_OTHER_ISSUE_12");
      end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "GUI_OTHER_ISSUE_12",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end GUI_OTHER_ISSUE_12_DRIVER;


  procedure GUI_FUEL_ON_HAND_124_DRIVER is
    LV_JET_QTY_AVAILABLE : INTEGER;
    LV_MOGAS_QTY_AVAILABLE : INTEGER;
    LV_DIESEL_QTY_AVAILABLE : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.

-- Data stream reads.
    begin

DS_JET_QTY_AVAILABLE_GUI_FUEL_ON_HAND_124.BUFFER.READ(LV_JET_QTY_AVAILAB
LE);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("JET_QTY_AVAILABLE_GUI_FUEL_ON_HAND_124",
"GUI_FUEL_ON_HAND_124");
    end;
    begin

DS_MOGAS_QTY_AVAILABLE_GUI_FUEL_ON_HAND_124.BUFFER.READ(LV_MOGAS_QTY_
AVAILABLE);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("MOGAS_QTY_AVAILABLE_GUI_FUEL_ON_HAND_124",
"GUI_FUEL_ON_HAND_124");
    end;
    begin

DS_DIESEL_QTY_AVAILABLE_GUI_FUEL_ON_HAND_124.BUFFER.READ(LV_DIESEL_QTY_A
VAILABLE);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("DIESEL_QTY_AVAILABLE_GUI_FUEL_ON_HAND_124",
"GUI_FUEL_ON_HAND_124");
    end;

-- Execution trigger condition check.
```

```
    if True then
     begin
     GUI_FUEL_ON_HAND_124(
       JET_QTY_AVAILABLE => LV_JET_QTY_AVAILABLE,
       MOGAS_QTY_AVAILABLE => LV_MOGAS_QTY_AVAILABLE,
       DIESEL_QTY_AVAILABLE => LV_DIESEL_QTY_AVAILABLE);
     exception
       when others =>
         DS_DEBUG.UNDECLARED_EXCEPTION("GUI_FUEL_ON_HAND_124");
         EXCEPTION_HAS_OCCURRED := true;
         EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
     end;
    else return;
    end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "GUI_FUEL_ON_HAND_124",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end GUI_FUEL_ON_HAND_124_DRIVER;


  procedure GUI_ACC_OFFICER_179_DRIVER is
    LV_TOLERANCE_DF : BOOLEAN;
    LV_TOLERANCE_MG : BOOLEAN;
    LV_TOLERANCE_JET : BOOLEAN;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.

-- Data stream reads.
    begin
      DS_TOLERANCE_DF_GUI_ACC_OFFICER_179.BUFFER.READ(LV_TOLERANCE_DF);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("TOLERANCE_DF_GUI_ACC_OFFICER_179",
"GUI_ACC_OFFICER_179");
    end;
    begin
      DS_TOLERANCE_MG_GUI_ACC_OFFICER_179.BUFFER.READ(LV_TOLERANCE_MG);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("TOLERANCE_MG_GUI_ACC_OFFICER_179",
"GUI_ACC_OFFICER_179");
    end;
    begin
```

```
      DS_TOLERANCE_JET_GUI_ACC_OFFICER_179.BUFFER.READ(LV_TOLERANCE_JET);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("TOLERANCE_JET_GUI_ACC_OFFICER_179",
"GUI_ACC_OFFICER_179");
    end;

-- Execution trigger condition check.
    if True then
      begin
      GUI_ACC_OFFICER_179(
        TOLERANCE_DF => LV_TOLERANCE_DF,
        TOLERANCE_MG => LV_TOLERANCE_MG,
        TOLERANCE_JET => LV_TOLERANCE_JET);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("GUI_ACC_OFFICER_179");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.

 -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "GUI_ACC_OFFICER_179",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end GUI_ACC_OFFICER_179_DRIVER;


  procedure JET_GAGE_917_DRIVER is
    LV_JET_QTY_ON_HAND : INTEGER;
    LV_JET_QTY_AVAILABLE : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_JET_QTY_ON_HAND_JET_GAGE_917.NEW_DATA) then
      return;
    end if;

 -- Data stream reads.
    begin
      DS_JET_QTY_ON_HAND_JET_GAGE_917.BUFFER.READ(LV_JET_QTY_ON_HAND);
    exception
      when BUFFER_UNDERFLOW =>
```

```
      DS_DEBUG.BUFFER_UNDERFLOW("JET_QTY_ON_HAND_JET_GAGE_917",
"JET_GAGE_917");
    end;

-- Execution trigger condition check.
    if True then
      begin
      JET_GAGE_917(
        JET_QTY_ON_HAND => LV_JET_QTY_ON_HAND,
        JET_QTY_AVAILABLE => LV_JET_QTY_AVAILABLE);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("JET_GAGE_917");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_JET_QTY_AVAILABLE_JET_ACCT_CALC_958.BUFFER.WRITE(LV_JET_QTY_AVAILABLE)
;
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("JET_QTY_AVAILABLE_JET_ACCT_CALC_958",
"JET_GAGE_917");
      end;
      begin

DS_JET_QTY_AVAILABLE_GUI_FUEL_ON_HAND_124.BUFFER.WRITE(LV_JET_QTY_AVAILA
BLE);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("JET_QTY_AVAILABLE_GUI_FUEL_ON_HAND_124",
"JET_GAGE_917");
      end;
    end if;

 -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "JET_GAGE_917",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end JET_GAGE_917_DRIVER;


  procedure JET_SUBTRACTION_914_DRIVER is
    LV_JET_ISS_QTY : INTEGER;
```

```
    LV_JET_QTY_ON_HAND : INTEGER;
    LV_JET_VOLUME : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
   begin
-- Data trigger checks.
    if not (DS_JET_ISS_QTY_JET_SUBTRACTION_914.NEW_DATA) then
     return;
    end if;

-- Data stream reads.
    begin
     DS_JET_VOLUME_JET_SUBTRACTION_914.BUFFER.READ(LV_JET_VOLUME);
    exception
     when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("JET_VOLUME_JET_SUBTRACTION_914",
"JET_SUBTRACTION_914");
    end;
    begin
     DS_JET_ISS_QTY_JET_SUBTRACTION_914.BUFFER.READ(LV_JET_ISS_QTY);
    exception
     when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("JET_ISS_QTY_JET_SUBTRACTION_914",
"JET_SUBTRACTION_914");
    end;

-- Execution trigger condition check.
    if True then
     begin
     JET_SUBTRACTION_914(
      JET_ISS_QTY => LV_JET_ISS_QTY,
      JET_QTY_ON_HAND => LV_JET_QTY_ON_HAND,
      JET_VOLUME => LV_JET_VOLUME);
     exception
      when others =>
       DS_DEBUG.UNDECLARED_EXCEPTION("JET_SUBTRACTION_914");
       EXCEPTION_HAS_OCCURRED := true;
       EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
     end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
     begin
      DS_JET_QTY_ON_HAND_JET_GAGE_917.BUFFER.WRITE(LV_JET_QTY_ON_HAND);
     exception
      when BUFFER_OVERFLOW =>
       DS_DEBUG.BUFFER_OVERFLOW("JET_QTY_ON_HAND_JET_GAGE_917",
 "JET_SUBTRACTION_914");
     end;
```

```
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_JET_VOLUME_JET_SUBTRACTION_914.BUFFER.WRITE(LV_JET_VOLUME);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("JET_VOLUME_JET_SUBTRACTION_914",
"JET_SUBTRACTION_914");
      end;
      begin
        DS_JET_VOLUME_JET_ADDITION_911.BUFFER.WRITE(LV_JET_VOLUME);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("JET_VOLUME_JET_ADDITION_911",
"JET_SUBTRACTION_914");
      end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "JET_SUBTRACTION_914",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end JET_SUBTRACTION_914_DRIVER;


  procedure JET_ADDITION_911_DRIVER is
    LV_JET_RCPT_QTY : INTEGER;
    LV_JET_QTY_ON_HAND : INTEGER;
    LV_JET_VOLUME : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_JET_RCPT_QTY_JET_ADDITION_911.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin
      DS_JET_VOLUME_JET_ADDITION_911.BUFFER.READ(LV_JET_VOLUME);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("JET_VOLUME_JET_ADDITION_911",
"JET_ADDITION_911");
    end;
    begin
      DS_JET_RCPT_QTY_JET_ADDITION_911.BUFFER.READ(LV_JET_RCPT_QTY);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("JET_RCPT_QTY_JET_ADDITION_911",
"JET_ADDITION_911");
    end;
```

```
-- Execution trigger condition check.
    if True then
      begin
      JET_ADDITION_911(
        JET_RCPT_QTY => LV_JET_RCPT_QTY,
        JET_QTY_ON_HAND => LV_JET_QTY_ON_HAND,
        JET_VOLUME => LV_JET_VOLUME);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("JET_ADDITION_911");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_JET_QTY_ON_HAND_JET_GAGE_917.BUFFER.WRITE(LV_JET_QTY_ON_HAND);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("JET_QTY_ON_HAND_JET_GAGE_917",
"JET_ADDITION_911");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_JET_VOLUME_JET_SUBTRACTION_914.BUFFER.WRITE(LV_JET_VOLUME);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("JET_VOLUME_JET_SUBTRACTION_914",
"JET_ADDITION_911");
      end;
      begin
        DS_JET_VOLUME_JET_ADDITION_911.BUFFER.WRITE(LV_JET_VOLUME);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("JET_VOLUME_JET_ADDITION_911",
"JET_ADDITION_911");
      end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "JET_ADDITION_911",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end JET_ADDITION_911_DRIVER;
```

```
procedure MOGAS_ADDITION_888_DRIVER is
  LV_MOGAS_RCPT_QTY : INTEGER;
  LV_MOGAS_VOLUME : INTEGER;
  LV_MG_QTY_ON_HAND : INTEGER;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
 begin
-- Data trigger checks.
  if not (DS_MOGAS_RCPT_QTY_MOGAS_ADDITION_888.NEW_DATA) then
   return;
  end if;

-- Data stream reads.
  begin

DS_MOGAS_RCPT_QTY_MOGAS_ADDITION_888.BUFFER.READ(LV_MOGAS_RCPT_QTY);
  exception
   when BUFFER_UNDERFLOW =>
    DS_DEBUG.BUFFER_UNDERFLOW("MOGAS_RCPT_QTY_MOGAS_ADDITION_888",
"MOGAS_ADDITION_888");
  end;
  begin
   DS_MOGAS_VOLUME_MOGAS_ADDITION_888.BUFFER.READ(LV_MOGAS_VOLUME);
  exception
   when BUFFER_UNDERFLOW =>
    DS_DEBUG.BUFFER_UNDERFLOW("MOGAS_VOLUME_MOGAS_ADDITION_888",
"MOGAS_ADDITION_888");
  end;

-- Execution trigger condition check.
  if True then
   begin
   MOGAS_ADDITION_888(
    MOGAS_RCPT_QTY => LV_MOGAS_RCPT_QTY,
    MOGAS_VOLUME => LV_MOGAS_VOLUME,
    MG_QTY_ON_HAND => LV_MG_QTY_ON_HAND);
   exception
    when others =>
     DS_DEBUG.UNDECLARED_EXCEPTION("MOGAS_ADDITION_888");
     EXCEPTION_HAS_OCCURRED := true;
     EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
   end;
  else return;
  end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
   begin

 DS_MOGAS_VOLUME_MOGAS_SUBTRACTION_891.BUFFER.WRITE(LV_MOGAS_VOLUME);
   exception
```

```
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("MOGAS_VOLUME_MOGAS_SUBTRACTION_891",
"MOGAS_ADDITION_888");
      end;
      begin
        DS_MOGAS_VOLUME_MOGAS_ADDITION_888.BUFFER.WRITE(LV_MOGAS_VOLUME);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("MOGAS_VOLUME_MOGAS_ADDITION_888",
"MOGAS_ADDITION_888");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_MG_QTY_ON_HAND_MOGAS_GAGE_894.BUFFER.WRITE(LV_MG_QTY_ON_HAND);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("MG_QTY_ON_HAND_MOGAS_GAGE_894",
"MOGAS_ADDITION_888");
      end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "MOGAS_ADDITION_888",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end MOGAS_ADDITION_888_DRIVER;


  procedure MOGAS_SUBTRACTION_891_DRIVER is
    LV_MOGAS_ISS_QTY : INTEGER;
    LV_MOGAS_VOLUME : INTEGER;
    LV_MG_QTY_ON_HAND : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_MOGAS_ISS_QTY_MOGAS_SUBTRACTION_891.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin

DS_MOGAS_ISS_QTY_MOGAS_SUBTRACTION_891.BUFFER.READ(LV_MOGAS_ISS_QTY);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("MOGAS_ISS_QTY_MOGAS_SUBTRACTION_891",
"MOGAS_SUBTRACTION_891");
    end;
    begin

DS_MOGAS_VOLUME_MOGAS_SUBTRACTION_891.BUFFER.READ(LV_MOGAS_VOLUME);
```

```
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("MOGAS_VOLUME_MOGAS_SUBTRACTION_891",
"MOGAS_SUBTRACTION_891");
    end;

-- Execution trigger condition check.
    if True then
      begin
      MOGAS_SUBTRACTION_891(
        MOGAS_ISS_QTY => LV_MOGAS_ISS_QTY,
        MOGAS_VOLUME => LV_MOGAS_VOLUME,
        MG_QTY_ON_HAND => LV_MG_QTY_ON_HAND);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("MOGAS_SUBTRACTION_891");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_MOGAS_VOLUME_MOGAS_SUBTRACTION_891.BUFFER.WRITE(LV_MOGAS_VOLUME);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("MOGAS_VOLUME_MOGAS_SUBTRACTION_891",
"MOGAS_SUBTRACTION_891");
      end;
      begin
        DS_MOGAS_VOLUME_MOGAS_ADDITION_888.BUFFER.WRITE(LV_MOGAS_VOLUME);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("MOGAS_VOLUME_MOGAS_ADDITION_888",
"MOGAS_SUBTRACTION_891");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_MG_QTY_ON_HAND_MOGAS_GAGE_894.BUFFER.WRITE(LV_MG_QTY_ON_HAND);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("MG_QTY_ON_HAND_MOGAS_GAGE_894",
"MOGAS_SUBTRACTION_891");
      end;
    end if;

  -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
```

```
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "MOGAS_SUBTRACTION_891",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end MOGAS_SUBTRACTION_891_DRIVER;


  procedure MOGAS_GAGE_894_DRIVER is
    LV_MG_QTY_ON_HAND : INTEGER;
    LV_MOGAS_QTY_AVAILABLE : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_MG_QTY_ON_HAND_MOGAS_GAGE_894.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin
      DS_MG_QTY_ON_HAND_MOGAS_GAGE_894.BUFFER.READ(LV_MG_QTY_ON_HAND);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("MG_QTY_ON_HAND_MOGAS_GAGE_894",
"MOGAS_GAGE_894");
    end;

-- Execution trigger condition check.
    if True then
      begin
      MOGAS_GAGE_894(
        MG_QTY_ON_HAND => LV_MG_QTY_ON_HAND,
        MOGAS_QTY_AVAILABLE => LV_MOGAS_QTY_AVAILABLE);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("MOGAS_GAGE_894");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
     if not EXCEPTION_HAS_OCCURRED then
       begin

DS_MOGAS_QTY_AVAILABLE_MG_ACCT_CALC_955.BUFFER.WRITE(LV_MOGAS_QTY_AVA
ILABLE);
       exception
         when BUFFER_OVERFLOW =>
```

```
        DS_DEBUG.BUFFER_OVERFLOW("MOGAS_QTY_AVAILABLE_MG_ACCT_CALC_955",
"MOGAS_GAGE_894");
      end;
      begin

DS_MOGAS_QTY_AVAILABLE_GUI_FUEL_ON_HAND_124.BUFFER.WRITE(LV_MOGAS_QTY_
AVAILABLE);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MOGAS_QTY_AVAILABLE_GUI_FUEL_ON_HAND_124",
"MOGAS_GAGE_894");
      end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "MOGAS_GAGE_894",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end MOGAS_GAGE_894_DRIVER;


  procedure OTHER_ISS_DB_TABLE_504_DRIVER is
    LV_EQ_ISS_NAME : TEXT_STRING_PKG.TEXT_STRING;
    LV_EQ_ISS_UNIT : TEXT_STRING_PKG.TEXT_STRING;
    LV_EQ_ISS_ID : TEXT_STRING_PKG.TEXT_STRING;
    LV_EQ_ISS_QTY : INTEGER;
    LV_EQ_ISS_FUEL_TYPE : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_EQ_ISS_FUEL_TYPE_OTHER_ISS_DB_TABLE_504.NEW_DATA or else
          DS_EQ_ISS_QTY_OTHER_ISS_DB_TABLE_504.NEW_DATA or else
          DS_EQ_ISS_ID_OTHER_ISS_DB_TABLE_504.NEW_DATA or else
          DS_EQ_ISS_UNIT_OTHER_ISS_DB_TABLE_504.NEW_DATA or else
          DS_EQ_ISS_NAME_OTHER_ISS_DB_TABLE_504.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin
      DS_EQ_ISS_NAME_OTHER_ISS_DB_TABLE_504.BUFFER.READ(LV_EQ_ISS_NAME);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("EQ_ISS_NAME_OTHER_ISS_DB_TABLE_504",
"OTHER_ISS_DB_TABLE_504");
    end;
    begin
      DS_EQ_ISS_UNIT_OTHER_ISS_DB_TABLE_504.BUFFER.READ(LV_EQ_ISS_UNIT);
    exception
      when BUFFER_UNDERFLOW =>
```

```
      DS_DEBUG.BUFFER_UNDERFLOW("EQ_ISS_UNIT_OTHER_ISS_DB_TABLE_504",
"OTHER_ISS_DB_TABLE_504");
    end;
    begin
      DS_EQ_ISS_ID_OTHER_ISS_DB_TABLE_504.BUFFER.READ(LV_EQ_ISS_ID);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("EQ_ISS_ID_OTHER_ISS_DB_TABLE_504",
"OTHER_ISS_DB_TABLE_504");
    end;
    begin
      DS_EQ_ISS_QTY_OTHER_ISS_DB_TABLE_504.BUFFER.READ(LV_EQ_ISS_QTY);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("EQ_ISS_QTY_OTHER_ISS_DB_TABLE_504",
"OTHER_ISS_DB_TABLE_504");
    end;
    begin

DS_EQ_ISS_FUEL_TYPE_OTHER_ISS_DB_TABLE_504.BUFFER.READ(LV_EQ_ISS_FUEL_TYPE
);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("EQ_ISS_FUEL_TYPE_OTHER_ISS_DB_TABLE_504",
"OTHER_ISS_DB_TABLE_504");
    end;

-- Execution trigger condition check.
    if True then
      begin
      OTHER_ISS_DB_TABLE_504(
        EQ_ISS_NAME => LV_EQ_ISS_NAME,
        EQ_ISS_UNIT => LV_EQ_ISS_UNIT,
        EQ_ISS_ID => LV_EQ_ISS_ID,
        EQ_ISS_QTY => LV_EQ_ISS_QTY,
        EQ_ISS_FUEL_TYPE => LV_EQ_ISS_FUEL_TYPE);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("OTHER_ISS_DB_TABLE_504");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.

 -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "OTHER_ISS_DB_TABLE_504",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
```

```
  end if;
 end OTHER_ISS_DB_TABLE_504_DRIVER;


 procedure OTHER_RCPT_DB_TABLE_501_DRIVER is
  LV_OTH_RCPT_SOURCE_UNIT : TEXT_STRING_PKG.TEXT_STRING;
  LV_OTH_RCPT_SOURCE_ID : TEXT_STRING_PKG.TEXT_STRING;
  LV_OTH_RCPT_QTY : INTEGER;
  LV_OTH_RCPT_FUEL_TYPE : INTEGER;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
 begin
-- Data trigger checks.
  if not (DS_OTH_RCPT_FUEL_TYPE_OTHER_RCPT_DB_TABLE_501.NEW_DATA or else
      DS_OTH_RCPT_QTY_OTHER_RCPT_DB_TABLE_501.NEW_DATA or else
      DS_OTH_RCPT_SOURCE_ID_OTHER_RCPT_DB_TABLE_501.NEW_DATA or else
      DS_OTH_RCPT_SOURCE_UNIT_OTHER_RCPT_DB_TABLE_501.NEW_DATA) then
   return;
  end if;

-- Data stream reads.
  begin

DS_OTH_RCPT_SOURCE_UNIT_OTHER_RCPT_DB_TABLE_501.BUFFER.READ(LV_OTH_RCPT
_SOURCE_UNIT);
  exception
   when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("OTH_RCPT_SOURCE_UNIT_OTHER_RCPT_DB_TABLE_50
1", "OTHER_RCPT_DB_TABLE_501");
  end;
  begin

DS_OTH_RCPT_SOURCE_ID_OTHER_RCPT_DB_TABLE_501.BUFFER.READ(LV_OTH_RCPT_S
OURCE_ID);
  exception
   when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("OTH_RCPT_SOURCE_ID_OTHER_RCPT_DB_TABLE_501",
"OTHER_RCPT_DB_TABLE_501");
  end;
  begin

DS_OTH_RCPT_QTY_OTHER_RCPT_DB_TABLE_501.BUFFER.READ(LV_OTH_RCPT_QTY);
  exception
   when BUFFER_UNDERFLOW =>
    DS_DEBUG.BUFFER_UNDERFLOW("OTH_RCPT_QTY_OTHER_RCPT_DB_TABLE_501",
"OTHER_RCPT_DB_TABLE_501");
  end;
  begin

DS_OTH_RCPT_FUEL_TYPE_OTHER_RCPT_DB_TABLE_501.BUFFER.READ(LV_OTH_RCPT_F
UEL_TYPE);
  exception
   when BUFFER_UNDERFLOW =>
```

```
DS_DEBUG.BUFFER_UNDERFLOW("OTH_RCPT_FUEL_TYPE_OTHER_RCPT_DB_TABLE_501",
"OTHER_RCPT_DB_TABLE_501");
   end;

-- Execution trigger condition check.
   if True then
    begin
    OTHER_RCPT_DB_TABLE_501(
     OTH_RCPT_SOURCE_UNIT => LV_OTH_RCPT_SOURCE_UNIT,
     OTH_RCPT_SOURCE_ID => LV_OTH_RCPT_SOURCE_ID,
     OTH_RCPT_QTY => LV_OTH_RCPT_QTY,
     OTH_RCPT_FUEL_TYPE => LV_OTH_RCPT_FUEL_TYPE);
    exception
     when others =>
      DS_DEBUG.UNDECLARED_EXCEPTION("OTHER_RCPT_DB_TABLE_501");
      EXCEPTION_HAS_OCCURRED := true;
      EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
   else return;
   end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.

 -- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
     "OTHER_RCPT_DB_TABLE_501",
     PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
   end if;
  end OTHER_RCPT_DB_TABLE_501_DRIVER;


  procedure BULK_ISS_DB_TABLE_498_DRIVER is
   LV_BULK_RCV_NAME : TEXT_STRING_PKG.TEXT_STRING;
   LV_BULK_RCV_UNIT : TEXT_STRING_PKG.TEXT_STRING;
   LV_BULK_ISS_DOC_NUM : TEXT_STRING_PKG.TEXT_STRING;
   LV_BULK_ISS_QTY : INTEGER;
   LV_BULK_ISS_FUEL_TYPE : INTEGER;

   EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
   EXCEPTION_ID: PSDL_EXCEPTION;
  begin
 -- Data trigger checks.
   if not (DS_BULK_ISS_FUEL_TYPE_BULK_ISS_DB_TABLE_498.NEW_DATA or else
       DS_BULK_ISS_QTY_BULK_ISS_DB_TABLE_498.NEW_DATA or else
       DS_BULK_ISS_DOC_NUM_BULK_ISS_DB_TABLE_498.NEW_DATA or else
       DS_BULK_RCV_UNIT_BULK_ISS_DB_TABLE_498.NEW_DATA or else
       DS_BULK_RCV_NAME_BULK_ISS_DB_TABLE_498.NEW_DATA) then
    return;
   end if;
```

```
-- Data stream reads.
    begin

DS_BULK_RCV_NAME_BULK_ISS_DB_TABLE_498.BUFFER.READ(LV_BULK_RCV_NAME);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("BULK_RCV_NAME_BULK_ISS_DB_TABLE_498",
"BULK_ISS_DB_TABLE_498");
    end;
    begin
      DS_BULK_RCV_UNIT_BULK_ISS_DB_TABLE_498.BUFFER.READ(LV_BULK_RCV_UNIT);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("BULK_RCV_UNIT_BULK_ISS_DB_TABLE_498",
"BULK_ISS_DB_TABLE_498");
    end;
    begin

DS_BULK_ISS_DOC_NUM_BULK_ISS_DB_TABLE_498.BUFFER.READ(LV_BULK_ISS_DOC_NU
M);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("BULK_ISS_DOC_NUM_BULK_ISS_DB_TABLE_498",
"BULK_ISS_DB_TABLE_498");
    end;
    begin
      DS_BULK_ISS_QTY_BULK_ISS_DB_TABLE_498.BUFFER.READ(LV_BULK_ISS_QTY);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("BULK_ISS_QTY_BULK_ISS_DB_TABLE_498",
"BULK_ISS_DB_TABLE_498");
    end;
    begin

DS_BULK_ISS_FUEL_TYPE_BULK_ISS_DB_TABLE_498.BUFFER.READ(LV_BULK_ISS_FUEL_T
YPE);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("BULK_ISS_FUEL_TYPE_BULK_ISS_DB_TABLE_498",
"BULK_ISS_DB_TABLE_498");
    end;

-- Execution trigger condition check.
    if True then
      begin
      BULK_ISS_DB_TABLE_498(
        BULK_RCV_NAME => LV_BULK_RCV_NAME,
        BULK_RCV_UNIT => LV_BULK_RCV_UNIT,
        BULK_ISS_DOC_NUM => LV_BULK_ISS_DOC_NUM,
        BULK_ISS_QTY => LV_BULK_ISS_QTY,
        BULK_ISS_FUEL_TYPE => LV_BULK_ISS_FUEL_TYPE);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("BULK_ISS_DB_TABLE_498");
          EXCEPTION_HAS_OCCURRED := true;
```

```
      EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
   else return;
   end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.

-- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
     DS_DEBUG.UNHANDLED_EXCEPTION(
       "BULK_ISS_DB_TABLE_498",
       PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
   end if;
  end BULK_ISS_DB_TABLE_498_DRIVER;


  procedure BULK_RCPT_DB_TABLE_495_DRIVER is
   LV_BULK_RCPT_DOC_NUMBER : TEXT_STRING_PKG.TEXT_STRING;
   LV_BULK_RCPT_QTY : INTEGER;
   LV_BULK_RCPT_FUEL_TYPE : INTEGER;

   EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
   EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
   if not (DS_BULK_RCPT_FUEL_TYPE_BULK_RCPT_DB_TABLE_495.NEW_DATA or else
        DS_BULK_RCPT_QTY_BULK_RCPT_DB_TABLE_495.NEW_DATA or else
        DS_BULK_RCPT_DOC_NUMBER_BULK_RCPT_DB_TABLE_495.NEW_DATA) then
     return;
   end if;

-- Data stream reads.
   begin

DS_BULK_RCPT_DOC_NUMBER_BULK_RCPT_DB_TABLE_495.BUFFER.READ(LV_BULK_RCP
T_DOC_NUMBER);
   exception
     when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("BULK_RCPT_DOC_NUMBER_BULK_RCPT_DB_TABLE_4
95", "BULK_RCPT_DB_TABLE_495");
   end;
   begin

DS_BULK_RCPT_QTY_BULK_RCPT_DB_TABLE_495.BUFFER.READ(LV_BULK_RCPT_QTY);
   exception
     when BUFFER_UNDERFLOW =>
       DS_DEBUG.BUFFER_UNDERFLOW("BULK_RCPT_QTY_BULK_RCPT_DB_TABLE_495",
"BULK_RCPT_DB_TABLE_495");
   end;
   begin
```

```
DS_BULK_RCPT_FUEL_TYPE_BULK_RCPT_DB_TABLE_495.BUFFER.READ(LV_BULK_RCPT_
FUEL_TYPE);
   exception
     when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("BULK_RCPT_FUEL_TYPE_BULK_RCPT_DB_TABLE_495",
"BULK_RCPT_DB_TABLE_495");
   end;

-- Execution trigger condition check.
   if True then
     begin
     BULK_RCPT_DB_TABLE_495(
       BULK_RCPT_DOC_NUMBER => LV_BULK_RCPT_DOC_NUMBER,
       BULK_RCPT_QTY => LV_BULK_RCPT_QTY,
       BULK_RCPT_FUEL_TYPE => LV_BULK_RCPT_FUEL_TYPE);
     exception
       when others =>
         DS_DEBUG.UNDECLARED_EXCEPTION("BULK_RCPT_DB_TABLE_495");
         EXCEPTION_HAS_OCCURRED := true;
         EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
     end;
   else return;
   end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.

 -- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
     DS_DEBUG.UNHANDLED_EXCEPTION(
       "BULK_RCPT_DB_TABLE_495",
       PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
   end if;
  end BULK_RCPT_DB_TABLE_495_DRIVER;


  procedure DAILY_REPORTER_410_DRIVER is
    LV_DAILY_JET_ISS_TOTAL : INTEGER;
    LV_DAILY_MG_ISS_TOTAL : INTEGER;
    LV_DAILY_DF_ISS_TOTAL : INTEGER;
    LV_DAILY_JET_RCPT_TOTAL : INTEGER;
    LV_DAILY_MG_RCPT_TOTAL : INTEGER;
    LV_DAILY_DF_RCPT_TOTAL : INTEGER;
    LV_MG_ISS_TOTAL : INTEGER;
    LV_DF_ISS_TOTAL : INTEGER;
    LV_JET_ISS_TOTAL : INTEGER;
    LV_JET_RCPT_TOTAL : INTEGER;
    LV_MG_RCPT_TOTAL : INTEGER;
    LV_DF_RCPT_TOTAL : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
```

```
    EXCEPTION_ID: PSDL_EXCEPTION;
   begin
-- Data trigger checks.

-- Data stream reads.
    begin
     DS_MG_ISS_TOTAL_DAILY_REPORTER_410.BUFFER.READ(LV_MG_ISS_TOTAL);
    exception
     when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("MG_ISS_TOTAL_DAILY_REPORTER_410",
"DAILY_REPORTER_410");
    end;
    begin
     DS_DF_ISS_TOTAL_DAILY_REPORTER_410.BUFFER.READ(LV_DF_ISS_TOTAL);
    exception
     when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("DF_ISS_TOTAL_DAILY_REPORTER_410",
"DAILY_REPORTER_410");
    end;
    begin
     DS_JET_ISS_TOTAL_DAILY_REPORTER_410.BUFFER.READ(LV_JET_ISS_TOTAL);
    exception
     when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("JET_ISS_TOTAL_DAILY_REPORTER_410",
"DAILY_REPORTER_410");
    end;
    begin
     DS_DF_RCPT_TOTAL_DAILY_REPORTER_410.BUFFER.READ(LV_DF_RCPT_TOTAL);
    exception
     when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("DF_RCPT_TOTAL_DAILY_REPORTER_410",
"DAILY_REPORTER_410");
    end;
    begin
     DS_MG_RCPT_TOTAL_DAILY_REPORTER_410.BUFFER.READ(LV_MG_RCPT_TOTAL);
    exception
     when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("MG_RCPT_TOTAL_DAILY_REPORTER_410",
"DAILY_REPORTER_410");
    end;
    begin
     DS_JET_RCPT_TOTAL_DAILY_REPORTER_410.BUFFER.READ(LV_JET_RCPT_TOTAL);
    exception
     when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("JET_RCPT_TOTAL_DAILY_REPORTER_410",
"DAILY_REPORTER_410");
    end;

-- Execution trigger condition check.
    if True then
     begin
     DAILY_REPORTER_410(
      DAILY_JET_ISS_TOTAL => LV_DAILY_JET_ISS_TOTAL,
      DAILY_MG_ISS_TOTAL => LV_DAILY_MG_ISS_TOTAL,
      DAILY_DF_ISS_TOTAL => LV_DAILY_DF_ISS_TOTAL,
      DAILY_JET_RCPT_TOTAL => LV_DAILY_JET_RCPT_TOTAL,
```

```
      DAILY_MG_RCPT_TOTAL => LV_DAILY_MG_RCPT_TOTAL,
      DAILY_DF_RCPT_TOTAL => LV_DAILY_DF_RCPT_TOTAL,
      MG_ISS_TOTAL => LV_MG_ISS_TOTAL,
      DF_ISS_TOTAL => LV_DF_ISS_TOTAL,
      JET_ISS_TOTAL => LV_JET_ISS_TOTAL,
      JET_RCPT_TOTAL => LV_JET_RCPT_TOTAL,
      MG_RCPT_TOTAL => LV_MG_RCPT_TOTAL,
      DF_RCPT_TOTAL => LV_DF_RCPT_TOTAL);
     exception
      when others =>
       DS_DEBUG.UNDECLARED_EXCEPTION("DAILY_REPORTER_410");
       EXCEPTION_HAS_OCCURRED := true;
       EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
     end;
    else return;
    end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
     begin

DS_DAILY_JET_ISS_TOTAL_MO_JET_ISS_TOTALIZER_598.BUFFER.WRITE(LV_DAILY_JET_IS
S_TOTAL);
     exception
      when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("DAILY_JET_ISS_TOTAL_MO_JET_ISS_TOTALIZER_598",
"DAILY_REPORTER_410");
     end;
     begin

DS_DAILY_JET_ISS_TOTAL_DAILY_ISS_DB_TABLE_743.BUFFER.WRITE(LV_DAILY_JET_ISS_
TOTAL);
     exception
      when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("DAILY_JET_ISS_TOTAL_DAILY_ISS_DB_TABLE_743",
"DAILY_REPORTER_410");
     end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
     begin

DS_DAILY_MG_ISS_TOTAL_MO_MG_ISS_TOTALIZER_595.BUFFER.WRITE(LV_DAILY_MG_IS
S_TOTAL);
     exception
      when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("DAILY_MG_ISS_TOTAL_MO_MG_ISS_TOTALIZER_595",
"DAILY_REPORTER_410");
     end;
     begin
```

```
DS_DAILY_MG_ISS_TOTAL_DAILY_ISS_DB_TABLE_743.BUFFER.WRITE(LV_DAILY_MG_ISS_
TOTAL);
    exception
      when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("DAILY_MG_ISS_TOTAL_DAILY_ISS_DB_TABLE_743",
"DAILY_REPORTER_410");
    end;
  end if;
  if not EXCEPTION_HAS_OCCURRED then
    begin

DS_DAILY_DF_ISS_TOTAL_MO_DF_ISS_TOTALIZER_592.BUFFER.WRITE(LV_DAILY_DF_ISS_
TOTAL);
    exception
      when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("DAILY_DF_ISS_TOTAL_MO_DF_ISS_TOTALIZER_592",
"DAILY_REPORTER_410");
    end;
    begin

DS_DAILY_DF_ISS_TOTAL_DAILY_ISS_DB_TABLE_743.BUFFER.WRITE(LV_DAILY_DF_ISS_T
OTAL);
    exception
      when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("DAILY_DF_ISS_TOTAL_DAILY_ISS_DB_TABLE_743",
"DAILY_REPORTER_410");
    end;
  end if;
  if not EXCEPTION_HAS_OCCURRED then
    begin

DS_DAILY_JET_RCPT_TOTAL_MO_JET_RCPT_TOTALIZER_589.BUFFER.WRITE(LV_DAILY_J
ET_RCPT_TOTAL);
    exception
      when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("DAILY_JET_RCPT_TOTAL_MO_JET_RCPT_TOTALIZER_58
9", "DAILY_REPORTER_410");
    end;
    begin

DS_DAILY_JET_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740.BUFFER.WRITE(LV_DAILY_JET
_RCPT_TOTAL);
    exception
      when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("DAILY_JET_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740",
"DAILY_REPORTER_410");
    end;
  end if;
  if not EXCEPTION_HAS_OCCURRED then
    begin
```

```
DS_DAILY_MG_RCPT_TOTAL_MO_MG_RCPT_TOTALIZER_586.BUFFER.WRITE(LV_DAILY_M
G_RCPT_TOTAL);
        exception
          when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("DAILY_MG_RCPT_TOTAL_MO_MG_RCPT_TOTALIZER_58
6", "DAILY_REPORTER_410");
        end;
        begin

DS_DAILY_MG_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740.BUFFER.WRITE(LV_DAILY_MG
_RCPT_TOTAL);
        exception
          when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("DAILY_MG_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740",
"DAILY_REPORTER_410");
        end;
      end if;
      if not EXCEPTION_HAS_OCCURRED then
        begin

DS_DAILY_DF_RCPT_TOTAL_MO_DF_RCPT_TOTALIZER_583.BUFFER.WRITE(LV_DAILY_DF
_RCPT_TOTAL);
        exception
          when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("DAILY_DF_RCPT_TOTAL_MO_DF_RCPT_TOTALIZER_583"
, "DAILY_REPORTER_410");
        end;
        begin

DS_DAILY_DF_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740.BUFFER.WRITE(LV_DAILY_DF_
RCPT_TOTAL);
        exception
          when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("DAILY_DF_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740",
"DAILY_REPORTER_410");
        end;
      end if;
      if not EXCEPTION_HAS_OCCURRED then
        begin
          DS_MG_ISS_TOTAL_MG_ISS_TOTALIZER_349.BUFFER.WRITE(LV_MG_ISS_TOTAL);
        exception
          when BUFFER_OVERFLOW =>
            DS_DEBUG.BUFFER_OVERFLOW("MG_ISS_TOTAL_MG_ISS_TOTALIZER_349",
"DAILY_REPORTER_410");
        end;
        begin
          DS_MG_ISS_TOTAL_DAILY_REPORTER_410.BUFFER.WRITE(LV_MG_ISS_TOTAL);
        exception
          when BUFFER_OVERFLOW =>
            DS_DEBUG.BUFFER_OVERFLOW("MG_ISS_TOTAL_DAILY_REPORTER_410",
"DAILY_REPORTER_410");
```

```
    end;
   end if;
   if not EXCEPTION_HAS_OCCURRED then
    begin
     DS_DF_ISS_TOTAL_DF_ISS_TOTALIZER_352.BUFFER.WRITE(LV_DF_ISS_TOTAL);
    exception
     when BUFFER_OVERFLOW =>
      DS_DEBUG.BUFFER_OVERFLOW("DF_ISS_TOTAL_DF_ISS_TOTALIZER_352",
"DAILY_REPORTER_410");
    end;
    begin
     DS_DF_ISS_TOTAL_DAILY_REPORTER_410.BUFFER.WRITE(LV_DF_ISS_TOTAL);
    exception
     when BUFFER_OVERFLOW =>
      DS_DEBUG.BUFFER_OVERFLOW("DF_ISS_TOTAL_DAILY_REPORTER_410",
"DAILY_REPORTER_410");
    end;
   end if;
   if not EXCEPTION_HAS_OCCURRED then
    begin
     DS_JET_ISS_TOTAL_JET_ISS_TOTALIZER_346.BUFFER.WRITE(LV_JET_ISS_TOTAL);
    exception
     when BUFFER_OVERFLOW =>
      DS_DEBUG.BUFFER_OVERFLOW("JET_ISS_TOTAL_JET_ISS_TOTALIZER_346",
"DAILY_REPORTER_410");
    end;
    begin
     DS_JET_ISS_TOTAL_DAILY_REPORTER_410.BUFFER.WRITE(LV_JET_ISS_TOTAL);
    exception
     when BUFFER_OVERFLOW =>
      DS_DEBUG.BUFFER_OVERFLOW("JET_ISS_TOTAL_DAILY_REPORTER_410",
"DAILY_REPORTER_410");
    end;
   end if;
   if not EXCEPTION_HAS_OCCURRED then
    begin

DS_JET_RCPT_TOTAL_JET_RCPT_TOTALIZER_280.BUFFER.WRITE(LV_JET_RCPT_TOTAL);
    exception
     when BUFFER_OVERFLOW =>
      DS_DEBUG.BUFFER_OVERFLOW("JET_RCPT_TOTAL_JET_RCPT_TOTALIZER_280",
"DAILY_REPORTER_410");
    end;
    begin
     DS_JET_RCPT_TOTAL_DAILY_REPORTER_410.BUFFER.WRITE(LV_JET_RCPT_TOTAL);
    exception
     when BUFFER_OVERFLOW =>
      DS_DEBUG.BUFFER_OVERFLOW("JET_RCPT_TOTAL_DAILY_REPORTER_410",
"DAILY_REPORTER_410");
    end;
   end if;
   if not EXCEPTION_HAS_OCCURRED then
    begin

DS_MG_RCPT_TOTAL_MG_RCPT_TOTALIZER_277.BUFFER.WRITE(LV_MG_RCPT_TOTAL);
    exception
```

```
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("MG_RCPT_TOTAL_MG_RCPT_TOTALIZER_277",
"DAILY_REPORTER_410");
      end;
      begin
        DS_MG_RCPT_TOTAL_DAILY_REPORTER_410.BUFFER.WRITE(LV_MG_RCPT_TOTAL);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("MG_RCPT_TOTAL_DAILY_REPORTER_410",
"DAILY_REPORTER_410");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_DF_RCPT_TOTAL_DF_RCPT_TOTALIZER_274.BUFFER.WRITE(LV_DF_RCPT_TOTAL);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("DF_RCPT_TOTAL_DF_RCPT_TOTALIZER_274",
"DAILY_REPORTER_410");
      end;
      begin
        DS_DF_RCPT_TOTAL_DAILY_REPORTER_410.BUFFER.WRITE(LV_DF_RCPT_TOTAL);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("DF_RCPT_TOTAL_DAILY_REPORTER_410",
"DAILY_REPORTER_410");
      end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "DAILY_REPORTER_410",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end DAILY_REPORTER_410_DRIVER;


  procedure DF_ISS_TOTALIZER_352_DRIVER is
    LV_I_DF_QTY : INTEGER;
    LV_DF_ISS_TOTAL : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_I_DF_QTY_DF_ISS_TOTALIZER_352.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin
      DS_I_DF_QTY_DF_ISS_TOTALIZER_352.BUFFER.READ(LV_I_DF_QTY);
    exception
      when BUFFER_UNDERFLOW =>
```

```
       DS_DEBUG.BUFFER_UNDERFLOW("I_DF_QTY_DF_ISS_TOTALIZER_352",
"DF_ISS_TOTALIZER_352");
    end;
    begin
      DS_DF_ISS_TOTAL_DF_ISS_TOTALIZER_352.BUFFER.READ(LV_DF_ISS_TOTAL);
    exception
      when BUFFER_UNDERFLOW =>
       DS_DEBUG.BUFFER_UNDERFLOW("DF_ISS_TOTAL_DF_ISS_TOTALIZER_352",
"DF_ISS_TOTALIZER_352");
    end;

-- Execution trigger condition check.
    if True then
      begin
      DF_ISS_TOTALIZER_352(
        I_DF_QTY => LV_I_DF_QTY,
        DF_ISS_TOTAL => LV_DF_ISS_TOTAL);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("DF_ISS_TOTALIZER_352");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_DF_ISS_TOTAL_DF_ISS_TOTALIZER_352.BUFFER.WRITE(LV_DF_ISS_TOTAL);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("DF_ISS_TOTAL_DF_ISS_TOTALIZER_352",
"DF_ISS_TOTALIZER_352");
      end;
      begin
        DS_DF_ISS_TOTAL_DAILY_REPORTER_410.BUFFER.WRITE(LV_DF_ISS_TOTAL);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("DF_ISS_TOTAL_DAILY_REPORTER_410",
"DF_ISS_TOTALIZER_352");
      end;
    end if;

 -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "DF_ISS_TOTALIZER_352",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end DF_ISS_TOTALIZER_352_DRIVER;
```

```
procedure MG_ISS_TOTALIZER_349_DRIVER is
  LV_I_MG_QTY : INTEGER;
  LV_MG_ISS_TOTAL : INTEGER;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
 begin
-- Data trigger checks.
  if not (DS_I_MG_QTY_MG_ISS_TOTALIZER_349.NEW_DATA) then
   return;
  end if;

-- Data stream reads.
  begin
   DS_I_MG_QTY_MG_ISS_TOTALIZER_349.BUFFER.READ(LV_I_MG_QTY);
  exception
   when BUFFER_UNDERFLOW =>
    DS_DEBUG.BUFFER_UNDERFLOW("I_MG_QTY_MG_ISS_TOTALIZER_349",
"MG_ISS_TOTALIZER_349");
  end;
  begin
   DS_MG_ISS_TOTAL_MG_ISS_TOTALIZER_349.BUFFER.READ(LV_MG_ISS_TOTAL);
  exception
   when BUFFER_UNDERFLOW =>
    DS_DEBUG.BUFFER_UNDERFLOW("MG_ISS_TOTAL_MG_ISS_TOTALIZER_349",
"MG_ISS_TOTALIZER_349");
  end;

-- Execution trigger condition check.
  if True then
   begin
   MG_ISS_TOTALIZER_349(
    I_MG_QTY => LV_I_MG_QTY,
    MG_ISS_TOTAL => LV_MG_ISS_TOTAL);
   exception
    when others =>
     DS_DEBUG.UNDECLARED_EXCEPTION("MG_ISS_TOTALIZER_349");
     EXCEPTION_HAS_OCCURRED := true;
     EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
   end;
  else return;
  end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
   begin
    DS_MG_ISS_TOTAL_MG_ISS_TOTALIZER_349.BUFFER.WRITE(LV_MG_ISS_TOTAL);
   exception
    when BUFFER_OVERFLOW =>
     DS_DEBUG.BUFFER_OVERFLOW("MG_ISS_TOTAL_MG_ISS_TOTALIZER_349",
"MG_ISS_TOTALIZER_349");
```

```
      end;
      begin
        DS_MG_ISS_TOTAL_DAILY_REPORTER_410.BUFFER.WRITE(LV_MG_ISS_TOTAL);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("MG_ISS_TOTAL_DAILY_REPORTER_410",
"MG_ISS_TOTALIZER_349");
      end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "MG_ISS_TOTALIZER_349",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end MG_ISS_TOTALIZER_349_DRIVER;


  procedure JET_ISS_TOTALIZER_346_DRIVER is
    LV_I_JET_QTY : INTEGER;
    LV_JET_ISS_TOTAL : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_I_JET_QTY_JET_ISS_TOTALIZER_346.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin
      DS_I_JET_QTY_JET_ISS_TOTALIZER_346.BUFFER.READ(LV_I_JET_QTY);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("I_JET_QTY_JET_ISS_TOTALIZER_346",
"JET_ISS_TOTALIZER_346");
    end;
    begin
      DS_JET_ISS_TOTAL_JET_ISS_TOTALIZER_346.BUFFER.READ(LV_JET_ISS_TOTAL);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("JET_ISS_TOTAL_JET_ISS_TOTALIZER_346",
"JET_ISS_TOTALIZER_346");
    end;

-- Execution trigger condition check.
    if True then
      begin
      JET_ISS_TOTALIZER_346(
        I_JET_QTY => LV_I_JET_QTY,
        JET_ISS_TOTAL => LV_JET_ISS_TOTAL);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("JET_ISS_TOTALIZER_346");
```

```
      EXCEPTION_HAS_OCCURRED := true;
      EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
   else return;
   end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
   if not EXCEPTION_HAS_OCCURRED then
    begin
     DS_JET_ISS_TOTAL_JET_ISS_TOTALIZER_346.BUFFER.WRITE(LV_JET_ISS_TOTAL);
    exception
     when BUFFER_OVERFLOW =>
      DS_DEBUG.BUFFER_OVERFLOW("JET_ISS_TOTAL_JET_ISS_TOTALIZER_346",
"JET_ISS_TOTALIZER_346");
    end;
    begin
     DS_JET_ISS_TOTAL_DAILY_REPORTER_410.BUFFER.WRITE(LV_JET_ISS_TOTAL);
    exception
     when BUFFER_OVERFLOW =>
      DS_DEBUG.BUFFER_OVERFLOW("JET_ISS_TOTAL_DAILY_REPORTER_410",
"JET_ISS_TOTALIZER_346");
    end;
   end if;

-- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
     "JET_ISS_TOTALIZER_346",
     PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
   end if;
  end JET_ISS_TOTALIZER_346_DRIVER;


  procedure JET_RCPT_TOTALIZER_280_DRIVER is
   LV_R_JET_QTY : INTEGER;
   LV_JET_RCPT_TOTAL : INTEGER;

   EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
   EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
   if not (DS_R_JET_QTY_JET_RCPT_TOTALIZER_280.NEW_DATA) then
    return;
   end if;

-- Data stream reads.
   begin
    DS_R_JET_QTY_JET_RCPT_TOTALIZER_280.BUFFER.READ(LV_R_JET_QTY);
   exception
    when BUFFER_UNDERFLOW =>
     DS_DEBUG.BUFFER_UNDERFLOW("R_JET_QTY_JET_RCPT_TOTALIZER_280",
"JET_RCPT_TOTALIZER_280");
```

```
    end;
    begin

DS_JET_RCPT_TOTAL_JET_RCPT_TOTALIZER_280.BUFFER.READ(LV_JET_RCPT_TOTAL);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("JET_RCPT_TOTAL_JET_RCPT_TOTALIZER_280",
"JET_RCPT_TOTALIZER_280");
    end;

-- Execution trigger condition check.
    if True then
      begin
      JET_RCPT_TOTALIZER_280(
        R_JET_QTY => LV_R_JET_QTY,
        JET_RCPT_TOTAL => LV_JET_RCPT_TOTAL);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("JET_RCPT_TOTALIZER_280");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_JET_RCPT_TOTAL_JET_RCPT_TOTALIZER_280.BUFFER.WRITE(LV_JET_RCPT_TOTAL);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("JET_RCPT_TOTAL_JET_RCPT_TOTALIZER_280",
"JET_RCPT_TOTALIZER_280");
      end;
      begin
        DS_JET_RCPT_TOTAL_DAILY_REPORTER_410.BUFFER.WRITE(LV_JET_RCPT_TOTAL);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("JET_RCPT_TOTAL_DAILY_REPORTER_410",
"JET_RCPT_TOTALIZER_280");
      end;
    end if;

 -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "JET_RCPT_TOTALIZER_280",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end JET_RCPT_TOTALIZER_280_DRIVER;
```

```
procedure MG_RCPT_TOTALIZER_277_DRIVER is
  LV_R_MG_QTY : INTEGER;
  LV_MG_RCPT_TOTAL : INTEGER;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
 begin
-- Data trigger checks.
   if not (DS_R_MG_QTY_MG_RCPT_TOTALIZER_277.NEW_DATA) then
     return;
   end if;

-- Data stream reads.
   begin
     DS_R_MG_QTY_MG_RCPT_TOTALIZER_277.BUFFER.READ(LV_R_MG_QTY);
   exception
     when BUFFER_UNDERFLOW =>
       DS_DEBUG.BUFFER_UNDERFLOW("R_MG_QTY_MG_RCPT_TOTALIZER_277",
"MG_RCPT_TOTALIZER_277");
   end;
   begin

DS_MG_RCPT_TOTAL_MG_RCPT_TOTALIZER_277.BUFFER.READ(LV_MG_RCPT_TOTAL);
   exception
     when BUFFER_UNDERFLOW =>
       DS_DEBUG.BUFFER_UNDERFLOW("MG_RCPT_TOTAL_MG_RCPT_TOTALIZER_277",
"MG_RCPT_TOTALIZER_277");
   end;

-- Execution trigger condition check.
   if True then
     begin
     MG_RCPT_TOTALIZER_277(
       R_MG_QTY => LV_R_MG_QTY,
       MG_RCPT_TOTAL => LV_MG_RCPT_TOTAL);
     exception
       when others =>
         DS_DEBUG.UNDECLARED_EXCEPTION("MG_RCPT_TOTALIZER_277");
         EXCEPTION_HAS_OCCURRED := true;
         EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
     end;
   else return;
   end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
   if not EXCEPTION_HAS_OCCURRED then
     begin

 DS_MG_RCPT_TOTAL_MG_RCPT_TOTALIZER_277.BUFFER.WRITE(LV_MG_RCPT_TOTAL);
     exception
       when BUFFER_OVERFLOW =>
```

```
        DS_DEBUG.BUFFER_OVERFLOW("MG_RCPT_TOTAL_MG_RCPT_TOTALIZER_277",
"MG_RCPT_TOTALIZER_277");
      end;
      begin
        DS_MG_RCPT_TOTAL_DAILY_REPORTER_410.BUFFER.WRITE(LV_MG_RCPT_TOTAL);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("MG_RCPT_TOTAL_DAILY_REPORTER_410",
"MG_RCPT_TOTALIZER_277");
      end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "MG_RCPT_TOTALIZER_277",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end MG_RCPT_TOTALIZER_277_DRIVER;


  procedure DF_RCPT_TOTALIZER_274_DRIVER is
    LV_R_DF_QTY : INTEGER;
    LV_DF_RCPT_TOTAL : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_R_DF_QTY_DF_RCPT_TOTALIZER_274.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin
      DS_R_DF_QTY_DF_RCPT_TOTALIZER_274.BUFFER.READ(LV_R_DF_QTY);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("R_DF_QTY_DF_RCPT_TOTALIZER_274",
"DF_RCPT_TOTALIZER_274");
    end;
    begin
      DS_DF_RCPT_TOTAL_DF_RCPT_TOTALIZER_274.BUFFER.READ(LV_DF_RCPT_TOTAL);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("DF_RCPT_TOTAL_DF_RCPT_TOTALIZER_274",
"DF_RCPT_TOTALIZER_274");
    end;

-- Execution trigger condition check.
    if True then
      begin
      DF_RCPT_TOTALIZER_274(
        R_DF_QTY => LV_R_DF_QTY,
        DF_RCPT_TOTAL => LV_DF_RCPT_TOTAL);
      exception
```

```
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("DF_RCPT_TOTALIZER_274");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_DF_RCPT_TOTAL_DF_RCPT_TOTALIZER_274.BUFFER.WRITE(LV_DF_RCPT_TOTAL);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("DF_RCPT_TOTAL_DF_RCPT_TOTALIZER_274",
"DF_RCPT_TOTALIZER_274");
      end;
      begin
        DS_DF_RCPT_TOTAL_DAILY_REPORTER_410.BUFFER.WRITE(LV_DF_RCPT_TOTAL);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("DF_RCPT_TOTAL_DAILY_REPORTER_410",
"DF_RCPT_TOTALIZER_274");
      end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "DF_RCPT_TOTALIZER_274",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end DF_RCPT_TOTALIZER_274_DRIVER;


  procedure BULK_RCPT_PROCESSOR_198_DRIVER is
    LV_BULK_RCPT_FUEL_TYPE : INTEGER;
    LV_BULK_RCPT_QTY : INTEGER;
    LV_BULK_RCPT_ENABLE : BOOLEAN;
    LV_OTH_RCPT_ENABLE : BOOLEAN;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_BULK_RCPT_FUEL_TYPE_BULK_RCPT_PROCESSOR_198.NEW_DATA and then
        DS_BULK_RCPT_QTY_BULK_RCPT_PROCESSOR_198.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
```

```
    begin

DS_BULK_RCPT_FUEL_TYPE_BULK_RCPT_PROCESSOR_198.BUFFER.READ(LV_BULK_RCPT
_FUEL_TYPE);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("BULK_RCPT_FUEL_TYPE_BULK_RCPT_PROCESSOR_198
", "BULK_RCPT_PROCESSOR_198");
    end;
    begin

DS_BULK_RCPT_QTY_BULK_RCPT_PROCESSOR_198.BUFFER.READ(LV_BULK_RCPT_QTY);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("BULK_RCPT_QTY_BULK_RCPT_PROCESSOR_198",
"BULK_RCPT_PROCESSOR_198");
    end;
    begin

DS_OTH_RCPT_ENABLE_BULK_RCPT_PROCESSOR_198.BUFFER.READ(LV_OTH_RCPT_ENAB
LE);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("OTH_RCPT_ENABLE_BULK_RCPT_PROCESSOR_198",
"BULK_RCPT_PROCESSOR_198");
    end;

-- Execution trigger condition check.
    if True then
      begin
      BULK_RCPT_PROCESSOR_198(
        BULK_RCPT_FUEL_TYPE => LV_BULK_RCPT_FUEL_TYPE,
        BULK_RCPT_QTY => LV_BULK_RCPT_QTY,
        BULK_RCPT_ENABLE => LV_BULK_RCPT_ENABLE,
        OTH_RCPT_ENABLE => LV_OTH_RCPT_ENABLE);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("BULK_RCPT_PROCESSOR_198");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_BULK_RCPT_ENABLE_OTH_RCPT_PROCESSOR_207.BUFFER.WRITE(LV_BULK_RCPT_EN
ABLE);
```

```
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("BULK_RCPT_ENABLE_OTH_RCPT_PROCESSOR_207",
"BULK_RCPT_PROCESSOR_198");
      end;
      begin

DS_BULK_RCPT_ENABLE_RCPT_PROCESSOR_210.BUFFER.WRITE(LV_BULK_RCPT_ENABLE
);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("BULK_RCPT_ENABLE_RCPT_PROCESSOR_210",
"BULK_RCPT_PROCESSOR_198");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_OTH_RCPT_ENABLE_BULK_RCPT_PROCESSOR_198.BUFFER.WRITE(LV_OTH_RCPT_ENA
BLE);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("OTH_RCPT_ENABLE_BULK_RCPT_PROCESSOR_198",
"BULK_RCPT_PROCESSOR_198");
      end;
      begin

DS_OTH_RCPT_ENABLE_RCPT_PROCESSOR_210.BUFFER.WRITE(LV_OTH_RCPT_ENABLE);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("OTH_RCPT_ENABLE_RCPT_PROCESSOR_210",
"BULK_RCPT_PROCESSOR_198");
      end;
    end if;

 -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "BULK_RCPT_PROCESSOR_198",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end BULK_RCPT_PROCESSOR_198_DRIVER;


  procedure OTH_RCPT_PROCESSOR_207_DRIVER is
    LV_OTH_RCPT_QTY : INTEGER;
    LV_OTH_RCPT_FUEL_TYPE : INTEGER;
    LV_OTH_RCPT_ENABLE : BOOLEAN;
    LV_BULK_RCPT_ENABLE : BOOLEAN;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
 -- Data trigger checks.
```

```
    if not (DS_OTH_RCPT_FUEL_TYPE_OTH_RCPT_PROCESSOR_207.NEW_DATA and then
          DS_OTH_RCPT_QTY_OTH_RCPT_PROCESSOR_207.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin
      DS_OTH_RCPT_QTY_OTH_RCPT_PROCESSOR_207.BUFFER.READ(LV_OTH_RCPT_QTY);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("OTH_RCPT_QTY_OTH_RCPT_PROCESSOR_207",
"OTH_RCPT_PROCESSOR_207");
    end;
    begin

DS_OTH_RCPT_FUEL_TYPE_OTH_RCPT_PROCESSOR_207.BUFFER.READ(LV_OTH_RCPT_FU
EL_TYPE);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("OTH_RCPT_FUEL_TYPE_OTH_RCPT_PROCESSOR_207",
"OTH_RCPT_PROCESSOR_207");
    end;
    begin

DS_BULK_RCPT_ENABLE_OTH_RCPT_PROCESSOR_207.BUFFER.READ(LV_BULK_RCPT_ENA
BLE);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("BULK_RCPT_ENABLE_OTH_RCPT_PROCESSOR_207",
"OTH_RCPT_PROCESSOR_207");
    end;

-- Execution trigger condition check.
    if True then
      begin
      OTH_RCPT_PROCESSOR_207(
        OTH_RCPT_QTY => LV_OTH_RCPT_QTY,
        OTH_RCPT_FUEL_TYPE => LV_OTH_RCPT_FUEL_TYPE,
        OTH_RCPT_ENABLE => LV_OTH_RCPT_ENABLE,
        BULK_RCPT_ENABLE => LV_BULK_RCPT_ENABLE);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("OTH_RCPT_PROCESSOR_207");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

  -- Exception Constraint translations.

  -- Other constraint option translations.

  --Unconditional output translations.
```

```
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_OTH_RCPT_ENABLE_BULK_RCPT_PROCESSOR_198.BUFFER.WRITE(LV_OTH_RCPT_ENA
BLE);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("OTH_RCPT_ENABLE_BULK_RCPT_PROCESSOR_198",
"OTH_RCPT_PROCESSOR_207");
      end;
      begin

DS_OTH_RCPT_ENABLE_RCPT_PROCESSOR_210.BUFFER.WRITE(LV_OTH_RCPT_ENABLE);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("OTH_RCPT_ENABLE_RCPT_PROCESSOR_210",
"OTH_RCPT_PROCESSOR_207");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_BULK_RCPT_ENABLE_OTH_RCPT_PROCESSOR_207.BUFFER.WRITE(LV_BULK_RCPT_EN
ABLE);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("BULK_RCPT_ENABLE_OTH_RCPT_PROCESSOR_207",
"OTH_RCPT_PROCESSOR_207");
      end;
      begin

DS_BULK_RCPT_ENABLE_RCPT_PROCESSOR_210.BUFFER.WRITE(LV_BULK_RCPT_ENABLE
);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("BULK_RCPT_ENABLE_RCPT_PROCESSOR_210",
"OTH_RCPT_PROCESSOR_207");
      end;
    end if;

  -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "OTH_RCPT_PROCESSOR_207",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end OTH_RCPT_PROCESSOR_207_DRIVER;


  procedure RCPT_PROCESSOR_210_DRIVER is
    LV_OTH_RCPT_QTY : INTEGER;
    LV_OTH_RCPT_FUEL_TYPE : INTEGER;
    LV_BULK_RCPT_QTY : INTEGER;
    LV_BULK_RCPT_FUEL_TYPE : INTEGER;
```

```
    LV_OTH_RCPT_ENABLE : BOOLEAN;
    LV_BULK_RCPT_ENABLE : BOOLEAN;
    LV_R_JET_QTY : INTEGER;
    LV_R_MG_QTY : INTEGER;
    LV_R_DF_QTY : INTEGER;
    LV_JET_RCPT_QTY : INTEGER;
    LV_MOGAS_RCPT_QTY : INTEGER;
    LV_DIESEL_RCPT_QTY : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
   begin
-- Data trigger checks.
    if not (DS_BULK_RCPT_ENABLE_RCPT_PROCESSOR_210.NEW_DATA or else
        DS_OTH_RCPT_ENABLE_RCPT_PROCESSOR_210.NEW_DATA) then
     return;
    end if;

-- Data stream reads.
    begin
     DS_OTH_RCPT_QTY_RCPT_PROCESSOR_210.BUFFER.READ(LV_OTH_RCPT_QTY);
    exception
     when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("OTH_RCPT_QTY_RCPT_PROCESSOR_210",
"RCPT_PROCESSOR_210");
    end;
    begin

DS_OTH_RCPT_FUEL_TYPE_RCPT_PROCESSOR_210.BUFFER.READ(LV_OTH_RCPT_FUEL_TY
PE);
    exception
     when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("OTH_RCPT_FUEL_TYPE_RCPT_PROCESSOR_210",
"RCPT_PROCESSOR_210");
    end;
    begin
     DS_BULK_RCPT_QTY_RCPT_PROCESSOR_210.BUFFER.READ(LV_BULK_RCPT_QTY);
    exception
     when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("BULK_RCPT_QTY_RCPT_PROCESSOR_210",
"RCPT_PROCESSOR_210");
    end;
    begin

DS_BULK_RCPT_FUEL_TYPE_RCPT_PROCESSOR_210.BUFFER.READ(LV_BULK_RCPT_FUEL_
TYPE);
    exception
     when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("BULK_RCPT_FUEL_TYPE_RCPT_PROCESSOR_210",
"RCPT_PROCESSOR_210");
    end;
    begin

DS_OTH_RCPT_ENABLE_RCPT_PROCESSOR_210.BUFFER.READ(LV_OTH_RCPT_ENABLE);
    exception
     when BUFFER_UNDERFLOW =>
```

```
        DS_DEBUG.BUFFER_UNDERFLOW("OTH_RCPT_ENABLE_RCPT_PROCESSOR_210",
"RCPT_PROCESSOR_210");
    end;
    begin

DS_BULK_RCPT_ENABLE_RCPT_PROCESSOR_210.BUFFER.READ(LV_BULK_RCPT_ENABLE);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("BULK_RCPT_ENABLE_RCPT_PROCESSOR_210",
"RCPT_PROCESSOR_210");
    end;

-- Execution trigger condition check.
    if True then
      begin
      RCPT_PROCESSOR_210(
        OTH_RCPT_QTY => LV_OTH_RCPT_QTY,
        OTH_RCPT_FUEL_TYPE => LV_OTH_RCPT_FUEL_TYPE,
        BULK_RCPT_QTY => LV_BULK_RCPT_QTY,
        BULK_RCPT_FUEL_TYPE => LV_BULK_RCPT_FUEL_TYPE,
        OTH_RCPT_ENABLE => LV_OTH_RCPT_ENABLE,
        BULK_RCPT_ENABLE => LV_BULK_RCPT_ENABLE,
        R_JET_QTY => LV_R_JET_QTY,
        R_MG_QTY => LV_R_MG_QTY,
        R_DF_QTY => LV_R_DF_QTY,
        JET_RCPT_QTY => LV_JET_RCPT_QTY,
        MOGAS_RCPT_QTY => LV_MOGAS_RCPT_QTY,
        DIESEL_RCPT_QTY => LV_DIESEL_RCPT_QTY);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("RCPT_PROCESSOR_210");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_R_JET_QTY_JET_RCPT_TOTALIZER_280.BUFFER.WRITE(LV_R_JET_QTY);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("R_JET_QTY_JET_RCPT_TOTALIZER_280",
 "RCPT_PROCESSOR_210");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_R_MG_QTY_MG_RCPT_TOTALIZER_277.BUFFER.WRITE(LV_R_MG_QTY);
      exception
        when BUFFER_OVERFLOW =>
```

```
        DS_DEBUG.BUFFER_OVERFLOW("R_MG_QTY_MG_RCPT_TOTALIZER_277",
"RCPT_PROCESSOR_210");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_R_DF_QTY_DF_RCPT_TOTALIZER_274.BUFFER.WRITE(LV_R_DF_QTY);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("R_DF_QTY_DF_RCPT_TOTALIZER_274",
"RCPT_PROCESSOR_210");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_JET_RCPT_QTY_JET_ADDITION_911.BUFFER.WRITE(LV_JET_RCPT_QTY);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("JET_RCPT_QTY_JET_ADDITION_911",
"RCPT_PROCESSOR_210");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_MOGAS_RCPT_QTY_MOGAS_ADDITION_888.BUFFER.WRITE(LV_MOGAS_RCPT_QTY);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("MOGAS_RCPT_QTY_MOGAS_ADDITION_888",
"RCPT_PROCESSOR_210");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_DIESEL_RCPT_QTY_DIESEL_ADDITION_836.BUFFER.WRITE(LV_DIESEL_RCPT_QTY);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("DIESEL_RCPT_QTY_DIESEL_ADDITION_836",
"RCPT_PROCESSOR_210");
      end;
    end if;

  -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "RCPT_PROCESSOR_210",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end RCPT_PROCESSOR_210_DRIVER;


  procedure OTH_ISS_PROCESSOR_307_DRIVER is
    LV_EQ_ISS_QTY : INTEGER;
    LV_EQ_ISS_FUEL_TYPE : INTEGER;
    LV_OTH_ISS_ENABLE : BOOLEAN;
```

```
    LV_BULK_ISS_ENABLE : BOOLEAN;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_EQ_ISS_QTY_OTH_ISS_PROCESSOR_307.NEW_DATA and then
        DS_EQ_ISS_FUEL_TYPE_OTH_ISS_PROCESSOR_307.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin
      DS_EQ_ISS_QTY_OTH_ISS_PROCESSOR_307.BUFFER.READ(LV_EQ_ISS_QTY);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("EQ_ISS_QTY_OTH_ISS_PROCESSOR_307",
"OTH_ISS_PROCESSOR_307");
    end;
    begin

DS_EQ_ISS_FUEL_TYPE_OTH_ISS_PROCESSOR_307.BUFFER.READ(LV_EQ_ISS_FUEL_TYPE);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("EQ_ISS_FUEL_TYPE_OTH_ISS_PROCESSOR_307",
"OTH_ISS_PROCESSOR_307");
    end;
    begin

DS_BULK_ISS_ENABLE_OTH_ISS_PROCESSOR_307.BUFFER.READ(LV_BULK_ISS_ENABLE);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("BULK_ISS_ENABLE_OTH_ISS_PROCESSOR_307",
"OTH_ISS_PROCESSOR_307");
    end;

-- Execution trigger condition check.
    if True then
      begin
      OTH_ISS_PROCESSOR_307(
        EQ_ISS_QTY => LV_EQ_ISS_QTY,
        EQ_ISS_FUEL_TYPE => LV_EQ_ISS_FUEL_TYPE,
        OTH_ISS_ENABLE => LV_OTH_ISS_ENABLE,
        BULK_ISS_ENABLE => LV_BULK_ISS_ENABLE);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("OTH_ISS_PROCESSOR_307");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

  -- Exception Constraint translations.

  -- Other constraint option translations.
```

```
--Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_OTH_ISS_ENABLE_BULK_ISS_PROCESSOR_310.BUFFER.WRITE(LV_OTH_ISS_ENABLE);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("OTH_ISS_ENABLE_BULK_ISS_PROCESSOR_310",
"OTH_ISS_PROCESSOR_307");
      end;
      begin
        DS_OTH_ISS_ENABLE_ISS_PROCESSOR_323.BUFFER.WRITE(LV_OTH_ISS_ENABLE);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("OTH_ISS_ENABLE_ISS_PROCESSOR_323",
"OTH_ISS_PROCESSOR_307");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_BULK_ISS_ENABLE_OTH_ISS_PROCESSOR_307.BUFFER.WRITE(LV_BULK_ISS_ENABLE);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("BULK_ISS_ENABLE_OTH_ISS_PROCESSOR_307",
"OTH_ISS_PROCESSOR_307");
      end;
      begin
        DS_BULK_ISS_ENABLE_ISS_PROCESSOR_323.BUFFER.WRITE(LV_BULK_ISS_ENABLE);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("BULK_ISS_ENABLE_ISS_PROCESSOR_323",
"OTH_ISS_PROCESSOR_307");
      end;
    end if;

 -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "OTH_ISS_PROCESSOR_307",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end OTH_ISS_PROCESSOR_307_DRIVER;


  procedure BULK_ISS_PROCESSOR_310_DRIVER is
    LV_BULK_ISS_QTY : INTEGER;
    LV_BULK_ISS_FUEL_TYPE : INTEGER;
    LV_BULK_ISS_ENABLE : BOOLEAN;
    LV_OTH_ISS_ENABLE : BOOLEAN;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
 -- Data trigger checks.
```

```
    if not (DS_BULK_ISS_FUEL_TYPE_BULK_ISS_PROCESSOR_310.NEW_DATA and then
        DS_BULK_ISS_QTY_BULK_ISS_PROCESSOR_310.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin
      DS_BULK_ISS_QTY_BULK_ISS_PROCESSOR_310.BUFFER.READ(LV_BULK_ISS_QTY);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("BULK_ISS_QTY_BULK_ISS_PROCESSOR_310",
"BULK_ISS_PROCESSOR_310");
    end;
    begin

DS_BULK_ISS_FUEL_TYPE_BULK_ISS_PROCESSOR_310.BUFFER.READ(LV_BULK_ISS_FUEL_
TYPE);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("BULK_ISS_FUEL_TYPE_BULK_ISS_PROCESSOR_310",
"BULK_ISS_PROCESSOR_310");
    end;
    begin

DS_OTH_ISS_ENABLE_BULK_ISS_PROCESSOR_310.BUFFER.READ(LV_OTH_ISS_ENABLE);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("OTH_ISS_ENABLE_BULK_ISS_PROCESSOR_310",
"BULK_ISS_PROCESSOR_310");
    end;

-- Execution trigger condition check.
    if True then
      begin
      BULK_ISS_PROCESSOR_310(
        BULK_ISS_QTY => LV_BULK_ISS_QTY,
        BULK_ISS_FUEL_TYPE => LV_BULK_ISS_FUEL_TYPE,
        BULK_ISS_ENABLE => LV_BULK_ISS_ENABLE,
        OTH_ISS_ENABLE => LV_OTH_ISS_ENABLE);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("BULK_ISS_PROCESSOR_310");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

  -- Exception Constraint translations.

  -- Other constraint option translations.

  --Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin
```

```
DS_BULK_ISS_ENABLE_OTH_ISS_PROCESSOR_307.BUFFER.WRITE(LV_BULK_ISS_ENABLE);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("BULK_ISS_ENABLE_OTH_ISS_PROCESSOR_307",
"BULK_ISS_PROCESSOR_310");
      end;
      begin
        DS_BULK_ISS_ENABLE_ISS_PROCESSOR_323.BUFFER.WRITE(LV_BULK_ISS_ENABLE);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("BULK_ISS_ENABLE_ISS_PROCESSOR_323",
"BULK_ISS_PROCESSOR_310");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_OTH_ISS_ENABLE_BULK_ISS_PROCESSOR_310.BUFFER.WRITE(LV_OTH_ISS_ENABLE);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("OTH_ISS_ENABLE_BULK_ISS_PROCESSOR_310",
"BULK_ISS_PROCESSOR_310");
      end;
      begin
        DS_OTH_ISS_ENABLE_ISS_PROCESSOR_323.BUFFER.WRITE(LV_OTH_ISS_ENABLE);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("OTH_ISS_ENABLE_ISS_PROCESSOR_323",
"BULK_ISS_PROCESSOR_310");
      end;
    end if;

 -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "BULK_ISS_PROCESSOR_310",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end BULK_ISS_PROCESSOR_310_DRIVER;


  procedure ISS_PROCESSOR_323_DRIVER is
    LV_BULK_ISS_ENABLE : BOOLEAN;
    LV_OTH_ISS_ENABLE : BOOLEAN;
    LV_EQ_ISS_FUEL_TYPE : INTEGER;
    LV_EQ_ISS_QTY : INTEGER;
    LV_BULK_ISS_FUEL_TYPE : INTEGER;
    LV_BULK_ISS_QTY : INTEGER;
    LV_I_DF_QTY : INTEGER;
    LV_I_MG_QTY : INTEGER;
    LV_I_JET_QTY : INTEGER;
    LV_DIESEL_ISS_QTY : INTEGER;
    LV_MOGAS_ISS_QTY : INTEGER;
    LV_JET_ISS_QTY : INTEGER;
```

```
    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
   begin
-- Data trigger checks.
    if not (DS_BULK_ISS_ENABLE_ISS_PROCESSOR_323.NEW_DATA or else
         DS_OTH_ISS_ENABLE_ISS_PROCESSOR_323.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin
      DS_BULK_ISS_ENABLE_ISS_PROCESSOR_323.BUFFER.READ(LV_BULK_ISS_ENABLE);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("BULK_ISS_ENABLE_ISS_PROCESSOR_323",
"ISS_PROCESSOR_323");
    end;
    begin
      DS_OTH_ISS_ENABLE_ISS_PROCESSOR_323.BUFFER.READ(LV_OTH_ISS_ENABLE);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("OTH_ISS_ENABLE_ISS_PROCESSOR_323",
"ISS_PROCESSOR_323");
    end;
    begin
      DS_EQ_ISS_FUEL_TYPE_ISS_PROCESSOR_323.BUFFER.READ(LV_EQ_ISS_FUEL_TYPE);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("EQ_ISS_FUEL_TYPE_ISS_PROCESSOR_323",
"ISS_PROCESSOR_323");
    end;
    begin
      DS_EQ_ISS_QTY_ISS_PROCESSOR_323.BUFFER.READ(LV_EQ_ISS_QTY);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("EQ_ISS_QTY_ISS_PROCESSOR_323",
"ISS_PROCESSOR_323");
    end;
    begin

DS_BULK_ISS_FUEL_TYPE_ISS_PROCESSOR_323.BUFFER.READ(LV_BULK_ISS_FUEL_TYPE);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("BULK_ISS_FUEL_TYPE_ISS_PROCESSOR_323",
"ISS_PROCESSOR_323");
    end;
    begin
      DS_BULK_ISS_QTY_ISS_PROCESSOR_323.BUFFER.READ(LV_BULK_ISS_QTY);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("BULK_ISS_QTY_ISS_PROCESSOR_323",
"ISS_PROCESSOR_323");
    end;

-- Execution trigger condition check.
    if True then
```

```
   begin
   ISS_PROCESSOR_323(
     BULK_ISS_ENABLE => LV_BULK_ISS_ENABLE,
     OTH_ISS_ENABLE => LV_OTH_ISS_ENABLE,
     EQ_ISS_FUEL_TYPE => LV_EQ_ISS_FUEL_TYPE,
     EQ_ISS_QTY => LV_EQ_ISS_QTY,
     BULK_ISS_FUEL_TYPE => LV_BULK_ISS_FUEL_TYPE,
     BULK_ISS_QTY => LV_BULK_ISS_QTY,
     I_DF_QTY => LV_I_DF_QTY,
     I_MG_QTY => LV_I_MG_QTY,
     I_JET_QTY => LV_I_JET_QTY,
     DIESEL_ISS_QTY => LV_DIESEL_ISS_QTY,
     MOGAS_ISS_QTY => LV_MOGAS_ISS_QTY,
     JET_ISS_QTY => LV_JET_ISS_QTY);
   exception
     when others =>
       DS_DEBUG.UNDECLARED_EXCEPTION("ISS_PROCESSOR_323");
       EXCEPTION_HAS_OCCURRED := true;
       EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
   end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_I_DF_QTY_DF_ISS_TOTALIZER_352.BUFFER.WRITE(LV_I_DF_QTY);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("I_DF_QTY_DF_ISS_TOTALIZER_352",
"ISS_PROCESSOR_323");
    end;
  end if;
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_I_MG_QTY_MG_ISS_TOTALIZER_349.BUFFER.WRITE(LV_I_MG_QTY);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("I_MG_QTY_MG_ISS_TOTALIZER_349",
"ISS_PROCESSOR_323");
    end;
  end if;
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_I_JET_QTY_JET_ISS_TOTALIZER_346.BUFFER.WRITE(LV_I_JET_QTY);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("I_JET_QTY_JET_ISS_TOTALIZER_346",
"ISS_PROCESSOR_323");
    end;
  end if;
  if not EXCEPTION_HAS_OCCURRED then
```

```
      begin

DS_DIESEL_ISS_QTY_DIESEL_SUBTRACTION_839.BUFFER.WRITE(LV_DIESEL_ISS_QTY);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("DIESEL_ISS_QTY_DIESEL_SUBTRACTION_839",
"ISS_PROCESSOR_323");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_MOGAS_ISS_QTY_MOGAS_SUBTRACTION_891.BUFFER.WRITE(LV_MOGAS_ISS_QTY);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("MOGAS_ISS_QTY_MOGAS_SUBTRACTION_891",
"ISS_PROCESSOR_323");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_JET_ISS_QTY_JET_SUBTRACTION_914.BUFFER.WRITE(LV_JET_ISS_QTY);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("JET_ISS_QTY_JET_SUBTRACTION_914",
"ISS_PROCESSOR_323");
      end;
    end if;

 -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "ISS_PROCESSOR_323",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end ISS_PROCESSOR_323_DRIVER;


  procedure DAILY_ISS_DB_TABLE_743_DRIVER is
    LV_DAILY_DF_ISS_TOTAL : INTEGER;
    LV_DAILY_MG_ISS_TOTAL : INTEGER;
    LV_DAILY_JET_ISS_TOTAL : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
 -- Data trigger checks.
    if not (DS_DAILY_DF_ISS_TOTAL_DAILY_ISS_DB_TABLE_743.NEW_DATA or else
          DS_DAILY_MG_ISS_TOTAL_DAILY_ISS_DB_TABLE_743.NEW_DATA or else
          DS_DAILY_JET_ISS_TOTAL_DAILY_ISS_DB_TABLE_743.NEW_DATA) then
      return;
    end if;

 -- Data stream reads.
    begin
```

```
DS_DAILY_DF_ISS_TOTAL_DAILY_ISS_DB_TABLE_743.BUFFER.READ(LV_DAILY_DF_ISS_T
OTAL);
    exception
     when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("DAILY_DF_ISS_TOTAL_DAILY_ISS_DB_TABLE_743",
"DAILY_ISS_DB_TABLE_743");
    end;
    begin

DS_DAILY_MG_ISS_TOTAL_DAILY_ISS_DB_TABLE_743.BUFFER.READ(LV_DAILY_MG_ISS_
TOTAL);
    exception
     when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("DAILY_MG_ISS_TOTAL_DAILY_ISS_DB_TABLE_743",
"DAILY_ISS_DB_TABLE_743");
    end;
    begin

DS_DAILY_JET_ISS_TOTAL_DAILY_ISS_DB_TABLE_743.BUFFER.READ(LV_DAILY_JET_ISS_
TOTAL);
    exception
     when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("DAILY_JET_ISS_TOTAL_DAILY_ISS_DB_TABLE_743",
"DAILY_ISS_DB_TABLE_743");
    end;

-- Execution trigger condition check.
    if True then
     begin
     DAILY_ISS_DB_TABLE_743(
       DAILY_DF_ISS_TOTAL => LV_DAILY_DF_ISS_TOTAL,
       DAILY_MG_ISS_TOTAL => LV_DAILY_MG_ISS_TOTAL,
       DAILY_JET_ISS_TOTAL => LV_DAILY_JET_ISS_TOTAL);
     exception
       when others =>
         DS_DEBUG.UNDECLARED_EXCEPTION("DAILY_ISS_DB_TABLE_743");
         EXCEPTION_HAS_OCCURRED := true;
         EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
     end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.

 -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "DAILY_ISS_DB_TABLE_743",
```

```
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end DAILY_ISS_DB_TABLE_743_DRIVER;


  procedure DAILY_RCPT_DB_TABLE_740_DRIVER is
    LV_DAILY_DF_RCPT_TOTAL : INTEGER;
    LV_DAILY_MG_RCPT_TOTAL : INTEGER;
    LV_DAILY_JET_RCPT_TOTAL : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_DAILY_DF_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740.NEW_DATA or else
         DS_DAILY_MG_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740.NEW_DATA or else
         DS_DAILY_JET_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin

DS_DAILY_DF_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740.BUFFER.READ(LV_DAILY_DF_R
CPT_TOTAL);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("DAILY_DF_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740"
, "DAILY_RCPT_DB_TABLE_740");
    end;
    begin

DS_DAILY_MG_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740.BUFFER.READ(LV_DAILY_MG_
RCPT_TOTAL);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("DAILY_MG_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740
", "DAILY_RCPT_DB_TABLE_740");
    end;
    begin

DS_DAILY_JET_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740.BUFFER.READ(LV_DAILY_JET_
RCPT_TOTAL);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("DAILY_JET_RCPT_TOTAL_DAILY_RCPT_DB_TABLE_740
", "DAILY_RCPT_DB_TABLE_740");
    end;

-- Execution trigger condition check.
    if True then
      begin
      DAILY_RCPT_DB_TABLE_740(
```

```
      DAILY_DF_RCPT_TOTAL => LV_DAILY_DF_RCPT_TOTAL,
      DAILY_MG_RCPT_TOTAL => LV_DAILY_MG_RCPT_TOTAL,
      DAILY_JET_RCPT_TOTAL => LV_DAILY_JET_RCPT_TOTAL);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("DAILY_RCPT_DB_TABLE_740");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "DAILY_RCPT_DB_TABLE_740",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
 end DAILY_RCPT_DB_TABLE_740_DRIVER;


 procedure MONTHLY_REPORTER_601_DRIVER is
   LV_MO_ISS_JET_TOTAL : INTEGER;
   LV_MO_ISS_MG_TOTAL : INTEGER;
   LV_MO_ISS_DF_TOTAL : INTEGER;
   LV_MO_RCPT_JET_TOTAL : INTEGER;
   LV_MO_RCPT_MG_TOTAL : INTEGER;
   LV_MO_RCPT_DF_TOTAL : INTEGER;
   LV_MONTH_DF_ISS_TOTAL : INTEGER;
   LV_MONTH_DF_RCPT_TOTAL : INTEGER;
   LV_MONTH_MG_ISS_TOTAL : INTEGER;
   LV_MONTH_MG_RCPT_TOTAL : INTEGER;
   LV_MONTH_JET_RCPT_TOTAL : INTEGER;
   LV_MONTH_JET_ISS_TOTAL : INTEGER;

   EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
   EXCEPTION_ID: PSDL_EXCEPTION;
 begin
-- Data trigger checks.

-- Data stream reads.
    begin
DS_MO_ISS_JET_TOTAL_MONTHLY_REPORTER_601.BUFFER.READ(LV_MO_ISS_JET_TOTAL
);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("MO_ISS_JET_TOTAL_MONTHLY_REPORTER_601",
"MONTHLY_REPORTER_601");
    end;
```

```
    begin
DS_MO_ISS_MG_TOTAL_MONTHLY_REPORTER_601.BUFFER.READ(LV_MO_ISS_MG_TOTAL
);
    exception
     when BUFFER_UNDERFLOW =>
       DS_DEBUG.BUFFER_UNDERFLOW("MO_ISS_MG_TOTAL_MONTHLY_REPORTER_601",
"MONTHLY_REPORTER_601");
    end;
    begin
DS_MO_ISS_DF_TOTAL_MONTHLY_REPORTER_601.BUFFER.READ(LV_MO_ISS_DF_TOTAL);
    exception
     when BUFFER_UNDERFLOW =>
       DS_DEBUG.BUFFER_UNDERFLOW("MO_ISS_DF_TOTAL_MONTHLY_REPORTER_601",
"MONTHLY_REPORTER_601");
    end;
    begin
DS_MO_RCPT_JET_TOTAL_MONTHLY_REPORTER_601.BUFFER.READ(LV_MO_RCPT_JET_TO
TAL);
    exception
     when BUFFER_UNDERFLOW =>
DS_DEBUG.BUFFER_UNDERFLOW("MO_RCPT_JET_TOTAL_MONTHLY_REPORTER_601",
"MONTHLY_REPORTER_601");
    end;
    begin
DS_MO_RCPT_MG_TOTAL_MONTHLY_REPORTER_601.BUFFER.READ(LV_MO_RCPT_MG_TO
TAL);
    exception
     when BUFFER_UNDERFLOW =>
DS_DEBUG.BUFFER_UNDERFLOW("MO_RCPT_MG_TOTAL_MONTHLY_REPORTER_601",
"MONTHLY_REPORTER_601");
    end;
    begin
DS_MO_RCPT_DF_TOTAL_MONTHLY_REPORTER_601.BUFFER.READ(LV_MO_RCPT_DF_TOT
AL);
    exception
     when BUFFER_UNDERFLOW =>
DS_DEBUG.BUFFER_UNDERFLOW("MO_RCPT_DF_TOTAL_MONTHLY_REPORTER_601",
"MONTHLY_REPORTER_601");
    end;

-- Execution trigger condition check.
    if True then
     begin
     MONTHLY_REPORTER_601(
       MO_ISS_JET_TOTAL => LV_MO_ISS_JET_TOTAL,
       MO_ISS_MG_TOTAL => LV_MO_ISS_MG_TOTAL,
       MO_ISS_DF_TOTAL => LV_MO_ISS_DF_TOTAL,
       MO_RCPT_JET_TOTAL => LV_MO_RCPT_JET_TOTAL,
```

```
        MO_RCPT_MG_TOTAL => LV_MO_RCPT_MG_TOTAL,
        MO_RCPT_DF_TOTAL => LV_MO_RCPT_DF_TOTAL,
        MONTH_DF_ISS_TOTAL => LV_MONTH_DF_ISS_TOTAL,
        MONTH_DF_RCPT_TOTAL => LV_MONTH_DF_RCPT_TOTAL,
        MONTH_MG_ISS_TOTAL => LV_MONTH_MG_ISS_TOTAL,
        MONTH_MG_RCPT_TOTAL => LV_MONTH_MG_RCPT_TOTAL,
        MONTH_JET_RCPT_TOTAL => LV_MONTH_JET_RCPT_TOTAL,
        MONTH_JET_ISS_TOTAL => LV_MONTH_JET_ISS_TOTAL);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("MONTHLY_REPORTER_601");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_MO_ISS_JET_TOTAL_MO_JET_ISS_TOTALIZER_598.BUFFER.WRITE(LV_MO_ISS_JET_TOT
AL);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MO_ISS_JET_TOTAL_MO_JET_ISS_TOTALIZER_598",
"MONTHLY_REPORTER_601");
      end;
      begin

DS_MO_ISS_JET_TOTAL_MONTHLY_REPORTER_601.BUFFER.WRITE(LV_MO_ISS_JET_TOTA
L);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("MO_ISS_JET_TOTAL_MONTHLY_REPORTER_601",
"MONTHLY_REPORTER_601");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_MO_ISS_MG_TOTAL_MO_MG_ISS_TOTALIZER_595.BUFFER.WRITE(LV_MO_ISS_MG_TOT
AL);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MO_ISS_MG_TOTAL_MO_MG_ISS_TOTALIZER_595",
"MONTHLY_REPORTER_601");
      end;
      begin
```

```
DS_MO_ISS_MG_TOTAL_MONTHLY_REPORTER_601.BUFFER.WRITE(LV_MO_ISS_MG_TOTA
L);
       exception
         when BUFFER_OVERFLOW =>
           DS_DEBUG.BUFFER_OVERFLOW("MO_ISS_MG_TOTAL_MONTHLY_REPORTER_601",
"MONTHLY_REPORTER_601");
       end;
     end if;
     if not EXCEPTION_HAS_OCCURRED then
       begin

DS_MO_ISS_DF_TOTAL_MO_DF_ISS_TOTALIZER_592.BUFFER.WRITE(LV_MO_ISS_DF_TOTA
L);
       exception
         when BUFFER_OVERFLOW =>
           DS_DEBUG.BUFFER_OVERFLOW("MO_ISS_DF_TOTAL_MO_DF_ISS_TOTALIZER_592",
"MONTHLY_REPORTER_601");
       end;
       begin

DS_MO_ISS_DF_TOTAL_MONTHLY_REPORTER_601.BUFFER.WRITE(LV_MO_ISS_DF_TOTAL)
;
       exception
         when BUFFER_OVERFLOW =>
           DS_DEBUG.BUFFER_OVERFLOW("MO_ISS_DF_TOTAL_MONTHLY_REPORTER_601",
"MONTHLY_REPORTER_601");
       end;
     end if;
     if not EXCEPTION_HAS_OCCURRED then
       begin

DS_MO_RCPT_JET_TOTAL_MO_JET_RCPT_TOTALIZER_589.BUFFER.WRITE(LV_MO_RCPT_J
ET_TOTAL);
       exception
         when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MO_RCPT_JET_TOTAL_MO_JET_RCPT_TOTALIZER_589",
"MONTHLY_REPORTER_601");
       end;
       begin

DS_MO_RCPT_JET_TOTAL_MONTHLY_REPORTER_601.BUFFER.WRITE(LV_MO_RCPT_JET_T
OTAL);
       exception
         when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MO_RCPT_JET_TOTAL_MONTHLY_REPORTER_601",
"MONTHLY_REPORTER_601");
       end;
     end if;
     if not EXCEPTION_HAS_OCCURRED then
       begin

DS_MO_RCPT_MG_TOTAL_MO_MG_RCPT_TOTALIZER_586.BUFFER.WRITE(LV_MO_RCPT_M
G_TOTAL);
```

```
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MO_RCPT_MG_TOTAL_MO_MG_RCPT_TOTALIZER_586",
"MONTHLY_REPORTER_601");
      end;
      begin

DS_MO_RCPT_MG_TOTAL_MONTHLY_REPORTER_601.BUFFER.WRITE(LV_MO_RCPT_MG_T
OTAL);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MO_RCPT_MG_TOTAL_MONTHLY_REPORTER_601",
"MONTHLY_REPORTER_601");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_MO_RCPT_DF_TOTAL_MO_DF_RCPT_TOTALIZER_583.BUFFER.WRITE(LV_MO_RCPT_DF
_TOTAL);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MO_RCPT_DF_TOTAL_MO_DF_RCPT_TOTALIZER_583",
"MONTHLY_REPORTER_601");
      end;
      begin

DS_MO_RCPT_DF_TOTAL_MONTHLY_REPORTER_601.BUFFER.WRITE(LV_MO_RCPT_DF_TO
TAL);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("MO_RCPT_DF_TOTAL_MONTHLY_REPORTER_601",
"MONTHLY_REPORTER_601");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_MONTH_DF_ISS_TOTAL_DIESEL_ISS_ACCT_PROC_934.BUFFER.WRITE(LV_MONTH_DF_I
SS_TOTAL);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MONTH_DF_ISS_TOTAL_DIESEL_ISS_ACCT_PROC_934",
"MONTHLY_REPORTER_601");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_MONTH_DF_RCPT_TOTAL_DIESEL_RCPT_ACCT_PROC_937.BUFFER.WRITE(LV_MONTH_
DF_RCPT_TOTAL);
      exception
```

```
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MONTH_DF_RCPT_TOTAL_DIESEL_RCPT_ACCT_PROC_93
7", "MONTHLY_REPORTER_601");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_MONTH_MG_ISS_TOTAL_MOGAS_ISS_ACCT_PROC_940.BUFFER.WRITE(LV_MONTH_MG
_ISS_TOTAL);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MONTH_MG_ISS_TOTAL_MOGAS_ISS_ACCT_PROC_940",
"MONTHLY_REPORTER_601");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_MONTH_MG_RCPT_TOTAL_MOGAS_RCPT_ACCT_PROC_943.BUFFER.WRITE(LV_MONTH
_MG_RCPT_TOTAL);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MONTH_MG_RCPT_TOTAL_MOGAS_RCPT_ACCT_PROC_9
43", "MONTHLY_REPORTER_601");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_MONTH_JET_RCPT_TOTAL_JET_RCPT_ACCT_PROC_949.BUFFER.WRITE(LV_MONTH_JET
_RCPT_TOTAL);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MONTH_JET_RCPT_TOTAL_JET_RCPT_ACCT_PROC_949",
"MONTHLY_REPORTER_601");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_MONTH_JET_ISS_TOTAL_JET_ISS_ACCT_PROC_946.BUFFER.WRITE(LV_MONTH_JET_ISS
_TOTAL);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MONTH_JET_ISS_TOTAL_JET_ISS_ACCT_PROC_946",
"MONTHLY_REPORTER_601");
      end;
    end if;

 -- PSDL Exception handler.
```

```
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "MONTHLY_REPORTER_601",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end MONTHLY_REPORTER_601_DRIVER;


  procedure MO_JET_ISS_TOTALIZER_598_DRIVER is
    LV_DAILY_JET_ISS_TOTAL : INTEGER;
    LV_MO_ISS_JET_TOTAL : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_DAILY_JET_ISS_TOTAL_MO_JET_ISS_TOTALIZER_598.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin

DS_MO_ISS_JET_TOTAL_MO_JET_ISS_TOTALIZER_598.BUFFER.READ(LV_MO_ISS_JET_TOT
AL);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("MO_ISS_JET_TOTAL_MO_JET_ISS_TOTALIZER_598",
"MO_JET_ISS_TOTALIZER_598");
    end;
    begin

DS_DAILY_JET_ISS_TOTAL_MO_JET_ISS_TOTALIZER_598.BUFFER.READ(LV_DAILY_JET_IS
S_TOTAL);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("DAILY_JET_ISS_TOTAL_MO_JET_ISS_TOTALIZER_598",
"MO_JET_ISS_TOTALIZER_598");
    end;

-- Execution trigger condition check.
    if True then
      begin
      MO_JET_ISS_TOTALIZER_598(
        DAILY_JET_ISS_TOTAL => LV_DAILY_JET_ISS_TOTAL,
        MO_ISS_JET_TOTAL => LV_MO_ISS_JET_TOTAL);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("MO_JET_ISS_TOTALIZER_598");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;
```

```
-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_MO_ISS_JET_TOTAL_MO_JET_ISS_TOTALIZER_598.BUFFER.WRITE(LV_MO_ISS_JET_TOT
AL);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MO_ISS_JET_TOTAL_MO_JET_ISS_TOTALIZER_598",
"MO_JET_ISS_TOTALIZER_598");
      end;
      begin

DS_MO_ISS_JET_TOTAL_MONTHLY_REPORTER_601.BUFFER.WRITE(LV_MO_ISS_JET_TOTA
L);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("MO_ISS_JET_TOTAL_MONTHLY_REPORTER_601",
"MO_JET_ISS_TOTALIZER_598");
      end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "MO_JET_ISS_TOTALIZER_598",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end MO_JET_ISS_TOTALIZER_598_DRIVER;


  procedure MO_MG_ISS_TOTALIZER_595_DRIVER is
    LV_DAILY_MG_ISS_TOTAL : INTEGER;
    LV_MO_ISS_MG_TOTAL : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_DAILY_MG_ISS_TOTAL_MO_MG_ISS_TOTALIZER_595.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin

DS_MO_ISS_MG_TOTAL_MO_MG_ISS_TOTALIZER_595.BUFFER.READ(LV_MO_ISS_MG_TOT
AL);
    exception
      when BUFFER_UNDERFLOW =>
```

```
DS_DEBUG.BUFFER_UNDERFLOW("MO_ISS_MG_TOTAL_MO_MG_ISS_TOTALIZER_595",
"MO_MG_ISS_TOTALIZER_595");
   end;
   begin

DS_DAILY_MG_ISS_TOTAL_MO_MG_ISS_TOTALIZER_595.BUFFER.READ(LV_DAILY_MG_IS
S_TOTAL);
   exception
    when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("DAILY_MG_ISS_TOTAL_MO_MG_ISS_TOTALIZER_595",
"MO_MG_ISS_TOTALIZER_595");
   end;

-- Execution trigger condition check.
   if True then
    begin
    MO_MG_ISS_TOTALIZER_595(
     DAILY_MG_ISS_TOTAL => LV_DAILY_MG_ISS_TOTAL,
     MO_ISS_MG_TOTAL => LV_MO_ISS_MG_TOTAL);
    exception
     when others =>
      DS_DEBUG.UNDECLARED_EXCEPTION("MO_MG_ISS_TOTALIZER_595");
      EXCEPTION_HAS_OCCURRED := true;
      EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
   else return;
   end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
   if not EXCEPTION_HAS_OCCURRED then
    begin

DS_MO_ISS_MG_TOTAL_MO_MG_ISS_TOTALIZER_595.BUFFER.WRITE(LV_MO_ISS_MG_TOT
AL);
    exception
     when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MO_ISS_MG_TOTAL_MO_MG_ISS_TOTALIZER_595",
"MO_MG_ISS_TOTALIZER_595");
    end;
    begin

DS_MO_ISS_MG_TOTAL_MONTHLY_REPORTER_601.BUFFER.WRITE(LV_MO_ISS_MG_TOTA
L);
    exception
     when BUFFER_OVERFLOW =>
      DS_DEBUG.BUFFER_OVERFLOW("MO_ISS_MG_TOTAL_MONTHLY_REPORTER_601",
"MO_MG_ISS_TOTALIZER_595");
    end;
   end if;
```

```
-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "MO_MG_ISS_TOTALIZER_595",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end MO_MG_ISS_TOTALIZER_595_DRIVER;


  procedure MO_DF_ISS_TOTALIZER_592_DRIVER is
    LV_DAILY_DF_ISS_TOTAL : INTEGER;
    LV_MO_ISS_DF_TOTAL : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_DAILY_DF_ISS_TOTAL_MO_DF_ISS_TOTALIZER_592.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin

DS_MO_ISS_DF_TOTAL_MO_DF_ISS_TOTALIZER_592.BUFFER.READ(LV_MO_ISS_DF_TOTAL
);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("MO_ISS_DF_TOTAL_MO_DF_ISS_TOTALIZER_592",
"MO_DF_ISS_TOTALIZER_592");
    end;
    begin

DS_DAILY_DF_ISS_TOTAL_MO_DF_ISS_TOTALIZER_592.BUFFER.READ(LV_DAILY_DF_ISS_
TOTAL);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("DAILY_DF_ISS_TOTAL_MO_DF_ISS_TOTALIZER_592",
"MO_DF_ISS_TOTALIZER_592");
    end;

-- Execution trigger condition check.
    if True then
      begin
      MO_DF_ISS_TOTALIZER_592(
        DAILY_DF_ISS_TOTAL => LV_DAILY_DF_ISS_TOTAL,
        MO_ISS_DF_TOTAL => LV_MO_ISS_DF_TOTAL);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("MO_DF_ISS_TOTALIZER_592");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
```

```
    end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_MO_ISS_DF_TOTAL_MO_DF_ISS_TOTALIZER_592.BUFFER.WRITE(LV_MO_ISS_DF_TOTA
L);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("MO_ISS_DF_TOTAL_MO_DF_ISS_TOTALIZER_592",
"MO_DF_ISS_TOTALIZER_592");
      end;
      begin

DS_MO_ISS_DF_TOTAL_MONTHLY_REPORTER_601.BUFFER.WRITE(LV_MO_ISS_DF_TOTAL)
;
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("MO_ISS_DF_TOTAL_MONTHLY_REPORTER_601",
"MO_DF_ISS_TOTALIZER_592");
      end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "MO_DF_ISS_TOTALIZER_592",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end MO_DF_ISS_TOTALIZER_592_DRIVER;


  procedure MO_JET_RCPT_TOTALIZER_589_DRIVER is
    LV_DAILY_JET_RCPT_TOTAL : INTEGER;
    LV_MO_RCPT_JET_TOTAL : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_DAILY_JET_RCPT_TOTAL_MO_JET_RCPT_TOTALIZER_589.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin

DS_DAILY_JET_RCPT_TOTAL_MO_JET_RCPT_TOTALIZER_589.BUFFER.READ(LV_DAILY_JE
T_RCPT_TOTAL);
    exception
      when BUFFER_UNDERFLOW =>
```

```
DS_DEBUG.BUFFER_UNDERFLOW("DAILY_JET_RCPT_TOTAL_MO_JET_RCPT_TOTALIZER_5
89", "MO_JET_RCPT_TOTALIZER_589");
   end;
   begin

DS_MO_RCPT_JET_TOTAL_MO_JET_RCPT_TOTALIZER_589.BUFFER.READ(LV_MO_RCPT_JE
T_TOTAL);
   exception
    when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("MO_RCPT_JET_TOTAL_MO_JET_RCPT_TOTALIZER_589",
"MO_JET_RCPT_TOTALIZER_589");
   end;

-- Execution trigger condition check.
   if True then
    begin
    MO_JET_RCPT_TOTALIZER_589(
     DAILY_JET_RCPT_TOTAL => LV_DAILY_JET_RCPT_TOTAL,
     MO_RCPT_JET_TOTAL => LV_MO_RCPT_JET_TOTAL);
    exception
     when others =>
      DS_DEBUG.UNDECLARED_EXCEPTION("MO_JET_RCPT_TOTALIZER_589");
      EXCEPTION_HAS_OCCURRED := true;
      EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
   else return;
   end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
   if not EXCEPTION_HAS_OCCURRED then
    begin

DS_MO_RCPT_JET_TOTAL_MO_JET_RCPT_TOTALIZER_589.BUFFER.WRITE(LV_MO_RCPT_J
ET_TOTAL);
    exception
     when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MO_RCPT_JET_TOTAL_MO_JET_RCPT_TOTALIZER_589",
"MO_JET_RCPT_TOTALIZER_589");
    end;
    begin

DS_MO_RCPT_JET_TOTAL_MONTHLY_REPORTER_601.BUFFER.WRITE(LV_MO_RCPT_JET_T
OTAL);
    exception
     when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MO_RCPT_JET_TOTAL_MONTHLY_REPORTER_601",
"MO_JET_RCPT_TOTALIZER_589");
    end;
```

```
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "MO_JET_RCPT_TOTALIZER_589",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end MO_JET_RCPT_TOTALIZER_589_DRIVER;


  procedure MO_MG_RCPT_TOTALIZER_586_DRIVER is
    LV_DAILY_MG_RCPT_TOTAL : INTEGER;
    LV_MO_RCPT_MG_TOTAL : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_DAILY_MG_RCPT_TOTAL_MO_MG_RCPT_TOTALIZER_586.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin

DS_DAILY_MG_RCPT_TOTAL_MO_MG_RCPT_TOTALIZER_586.BUFFER.READ(LV_DAILY_M
G_RCPT_TOTAL);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("DAILY_MG_RCPT_TOTAL_MO_MG_RCPT_TOTALIZER_5
86", "MO_MG_RCPT_TOTALIZER_586");
    end;
    begin

DS_MO_RCPT_MG_TOTAL_MO_MG_RCPT_TOTALIZER_586.BUFFER.READ(LV_MO_RCPT_M
G_TOTAL);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("MO_RCPT_MG_TOTAL_MO_MG_RCPT_TOTALIZER_586",
"MO_MG_RCPT_TOTALIZER_586");
    end;

-- Execution trigger condition check.
    if True then
      begin
      MO_MG_RCPT_TOTALIZER_586(
        DAILY_MG_RCPT_TOTAL => LV_DAILY_MG_RCPT_TOTAL,
        MO_RCPT_MG_TOTAL => LV_MO_RCPT_MG_TOTAL);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("MO_MG_RCPT_TOTALIZER_586");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
```

```
      end;
     else return;
     end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
     if not EXCEPTION_HAS_OCCURRED then
       begin

DS_MO_RCPT_MG_TOTAL_MO_MG_RCPT_TOTALIZER_586.BUFFER.WRITE(LV_MO_RCPT_M
G_TOTAL);
       exception
         when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MO_RCPT_MG_TOTAL_MO_MG_RCPT_TOTALIZER_586",
"MO_MG_RCPT_TOTALIZER_586");
       end;
       begin

DS_MO_RCPT_MG_TOTAL_MONTHLY_REPORTER_601.BUFFER.WRITE(LV_MO_RCPT_MG_T
OTAL);
       exception
         when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MO_RCPT_MG_TOTAL_MONTHLY_REPORTER_601",
"MO_MG_RCPT_TOTALIZER_586");
       end;
     end if;

 -- PSDL Exception handler.
     if EXCEPTION_HAS_OCCURRED then
       DS_DEBUG.UNHANDLED_EXCEPTION(
         "MO_MG_RCPT_TOTALIZER_586",
         PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
     end if;
   end MO_MG_RCPT_TOTALIZER_586_DRIVER;


   procedure MO_DF_RCPT_TOTALIZER_583_DRIVER is
     LV_DAILY_DF_RCPT_TOTAL : INTEGER;
     LV_MO_RCPT_DF_TOTAL : INTEGER;

     EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
     EXCEPTION_ID: PSDL_EXCEPTION;
   begin
-- Data trigger checks.
     if not (DS_DAILY_DF_RCPT_TOTAL_MO_DF_RCPT_TOTALIZER_583.NEW_DATA) then
       return;
     end if;

-- Data stream reads.
     begin
```

```
DS_DAILY_DF_RCPT_TOTAL_MO_DF_RCPT_TOTALIZER_583.BUFFER.READ(LV_DAILY_DF_
RCPT_TOTAL);
    exception
     when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("DAILY_DF_RCPT_TOTAL_MO_DF_RCPT_TOTALIZER_58
3", "MO_DF_RCPT_TOTALIZER_583");
    end;
    begin

DS_MO_RCPT_DF_TOTAL_MO_DF_RCPT_TOTALIZER_583.BUFFER.READ(LV_MO_RCPT_DF_
TOTAL);
    exception
     when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("MO_RCPT_DF_TOTAL_MO_DF_RCPT_TOTALIZER_583",
"MO_DF_RCPT_TOTALIZER_583");
    end;

-- Execution trigger condition check.
    if True then
     begin
     MO_DF_RCPT_TOTALIZER_583(
      DAILY_DF_RCPT_TOTAL => LV_DAILY_DF_RCPT_TOTAL,
      MO_RCPT_DF_TOTAL => LV_MO_RCPT_DF_TOTAL);
     exception
      when others =>
       DS_DEBUG.UNDECLARED_EXCEPTION("MO_DF_RCPT_TOTALIZER_583");
       EXCEPTION_HAS_OCCURRED := true;
       EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
     end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
     begin

DS_MO_RCPT_DF_TOTAL_MO_DF_RCPT_TOTALIZER_583.BUFFER.WRITE(LV_MO_RCPT_DF
_TOTAL);
     exception
      when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("MO_RCPT_DF_TOTAL_MO_DF_RCPT_TOTALIZER_583",
"MO_DF_RCPT_TOTALIZER_583");
     end;
     begin

DS_MO_RCPT_DF_TOTAL_MONTHLY_REPORTER_601.BUFFER.WRITE(LV_MO_RCPT_DF_TO
TAL);
     exception
```

```
      when BUFFER_OVERFLOW =>
       DS_DEBUG.BUFFER_OVERFLOW("MO_RCPT_DF_TOTAL_MONTHLY_REPORTER_601",
"MO_DF_RCPT_TOTALIZER_583");
     end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
     DS_DEBUG.UNHANDLED_EXCEPTION(
      "MO_DF_RCPT_TOTALIZER_583",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end MO_DF_RCPT_TOTALIZER_583_DRIVER;


  procedure DIESEL_ISS_ACCT_PROC_934_DRIVER is
    LV_MONTH_DF_ISS_TOTAL : INTEGER;
    LV_TOTAL_MO_DF_ISS : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_MONTH_DF_ISS_TOTAL_DIESEL_ISS_ACCT_PROC_934.NEW_DATA) then
     return;
    end if;

-- Data stream reads.
    begin

DS_MONTH_DF_ISS_TOTAL_DIESEL_ISS_ACCT_PROC_934.BUFFER.READ(LV_MONTH_DF_IS
S_TOTAL);
    exception
     when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("MONTH_DF_ISS_TOTAL_DIESEL_ISS_ACCT_PROC_934",
"DIESEL_ISS_ACCT_PROC_934");
    end;

-- Execution trigger condition check.
    if True then
     begin
     DIESEL_ISS_ACCT_PROC_934(
       MONTH_DF_ISS_TOTAL => LV_MONTH_DF_ISS_TOTAL,
       TOTAL_MO_DF_ISS => LV_TOTAL_MO_DF_ISS);
     exception
       when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("DIESEL_ISS_ACCT_PROC_934");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
     end;
    else return;
    end if;

 -- Exception Constraint translations.
```

```
-- Other constraint option translations.

--Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_TOTAL_MO_DF_ISS_DF_ACCT_CALC_952.BUFFER.WRITE(LV_TOTAL_MO_DF_ISS);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("TOTAL_MO_DF_ISS_DF_ACCT_CALC_952",
"DIESEL_ISS_ACCT_PROC_934");
      end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "DIESEL_ISS_ACCT_PROC_934",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end DIESEL_ISS_ACCT_PROC_934_DRIVER;


  procedure DIESEL_RCPT_ACCT_PROC_937_DRIVER is
    LV_MONTH_DF_RCPT_TOTAL : INTEGER;
    LV_TOTAL_MO_DF_RCPT : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_MONTH_DF_RCPT_TOTAL_DIESEL_RCPT_ACCT_PROC_937.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin

DS_MONTH_DF_RCPT_TOTAL_DIESEL_RCPT_ACCT_PROC_937.BUFFER.READ(LV_MONTH_
DF_RCPT_TOTAL);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("MONTH_DF_RCPT_TOTAL_DIESEL_RCPT_ACCT_PROC_
937", "DIESEL_RCPT_ACCT_PROC_937");
    end;

-- Execution trigger condition check.
    if True then
      begin
      DIESEL_RCPT_ACCT_PROC_937(
        MONTH_DF_RCPT_TOTAL => LV_MONTH_DF_RCPT_TOTAL,
        TOTAL_MO_DF_RCPT => LV_TOTAL_MO_DF_RCPT);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("DIESEL_RCPT_ACCT_PROC_937");
          EXCEPTION_HAS_OCCURRED := true;
```

```
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
        end;
      else return;
      end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
      if not EXCEPTION_HAS_OCCURRED then
        begin

DS_TOTAL_MO_DF_RCPT_DF_ACCT_CALC_952.BUFFER.WRITE(LV_TOTAL_MO_DF_RCPT);
        exception
          when BUFFER_OVERFLOW =>
            DS_DEBUG.BUFFER_OVERFLOW("TOTAL_MO_DF_RCPT_DF_ACCT_CALC_952",
"DIESEL_RCPT_ACCT_PROC_937");
        end;
      end if;

-- PSDL Exception handler.
      if EXCEPTION_HAS_OCCURRED then
        DS_DEBUG.UNHANDLED_EXCEPTION(
          "DIESEL_RCPT_ACCT_PROC_937",
          PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
      end if;
    end DIESEL_RCPT_ACCT_PROC_937_DRIVER;


    procedure MOGAS_ISS_ACCT_PROC_940_DRIVER is
      LV_MONTH_MG_ISS_TOTAL : INTEGER;
      LV_TOTAL_MO_MG_ISS : INTEGER;

      EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
      EXCEPTION_ID: PSDL_EXCEPTION;
    begin
-- Data trigger checks.
      if not (DS_MONTH_MG_ISS_TOTAL_MOGAS_ISS_ACCT_PROC_940.NEW_DATA) then
        return;
      end if;

-- Data stream reads.
      begin

DS_MONTH_MG_ISS_TOTAL_MOGAS_ISS_ACCT_PROC_940.BUFFER.READ(LV_MONTH_MG_
ISS_TOTAL);
      exception
        when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("MONTH_MG_ISS_TOTAL_MOGAS_ISS_ACCT_PROC_940"
, "MOGAS_ISS_ACCT_PROC_940");
      end;

-- Execution trigger condition check.
      if True then
```

```
      begin
      MOGAS_ISS_ACCT_PROC_940(
        MONTH_MG_ISS_TOTAL => LV_MONTH_MG_ISS_TOTAL,
        TOTAL_MO_MG_ISS => LV_TOTAL_MO_MG_ISS);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("MOGAS_ISS_ACCT_PROC_940");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_TOTAL_MO_MG_ISS_MG_ACCT_CALC_955.BUFFER.WRITE(LV_TOTAL_MO_MG_ISS);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("TOTAL_MO_MG_ISS_MG_ACCT_CALC_955",
"MOGAS_ISS_ACCT_PROC_940");
      end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "MOGAS_ISS_ACCT_PROC_940",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end MOGAS_ISS_ACCT_PROC_940_DRIVER;


  procedure MOGAS_RCPT_ACCT_PROC_943_DRIVER is
    LV_MONTH_MG_RCPT_TOTAL : INTEGER;
    LV_TOTAL_MO_MG_RCPT : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_MONTH_MG_RCPT_TOTAL_MOGAS_RCPT_ACCT_PROC_943.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin

DS_MONTH_MG_RCPT_TOTAL_MOGAS_RCPT_ACCT_PROC_943.BUFFER.READ(LV_MONTH_
MG_RCPT_TOTAL);
    exception
```

```
    when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("MONTH_MG_RCPT_TOTAL_MOGAS_RCPT_ACCT_PROC
_943", "MOGAS_RCPT_ACCT_PROC_943");
   end;

-- Execution trigger condition check.
   if True then
    begin
    MOGAS_RCPT_ACCT_PROC_943(
     MONTH_MG_RCPT_TOTAL => LV_MONTH_MG_RCPT_TOTAL,
     TOTAL_MO_MG_RCPT => LV_TOTAL_MO_MG_RCPT);
    exception
     when others =>
      DS_DEBUG.UNDECLARED_EXCEPTION("MOGAS_RCPT_ACCT_PROC_943");
      EXCEPTION_HAS_OCCURRED := true;
      EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
   else return;
   end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
   if not EXCEPTION_HAS_OCCURRED then
    begin

DS_TOTAL_MO_MG_RCPT_MG_ACCT_CALC_955.BUFFER.WRITE(LV_TOTAL_MO_MG_RCPT)
;
    exception
     when BUFFER_OVERFLOW =>
      DS_DEBUG.BUFFER_OVERFLOW("TOTAL_MO_MG_RCPT_MG_ACCT_CALC_955",
"MOGAS_RCPT_ACCT_PROC_943");
    end;
   end if;

 -- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
     "MOGAS_RCPT_ACCT_PROC_943",
     PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
   end if;
  end MOGAS_RCPT_ACCT_PROC_943_DRIVER;


  procedure JET_ISS_ACCT_PROC_946_DRIVER is
   LV_MONTH_JET_ISS_TOTAL : INTEGER;
   LV_TOTAL_MO_JET_ISS : INTEGER;

   EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
   EXCEPTION_ID: PSDL_EXCEPTION;
  begin
 -- Data trigger checks.
   if not (DS_MONTH_JET_ISS_TOTAL_JET_ISS_ACCT_PROC_946.NEW_DATA) then
```

```
      return;
    end if;

-- Data stream reads.
    begin

DS_MONTH_JET_ISS_TOTAL_JET_ISS_ACCT_PROC_946.BUFFER.READ(LV_MONTH_JET_ISS_
TOTAL);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("MONTH_JET_ISS_TOTAL_JET_ISS_ACCT_PROC_946",
"JET_ISS_ACCT_PROC_946");
    end;

-- Execution trigger condition check.
    if True then
      begin
      JET_ISS_ACCT_PROC_946(
        MONTH_JET_ISS_TOTAL => LV_MONTH_JET_ISS_TOTAL,
        TOTAL_MO_JET_ISS => LV_TOTAL_MO_JET_ISS);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("JET_ISS_ACCT_PROC_946");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_TOTAL_MO_JET_ISS_JET_ACCT_CALC_958.BUFFER.WRITE(LV_TOTAL_MO_JET_ISS);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("TOTAL_MO_JET_ISS_JET_ACCT_CALC_958",
"JET_ISS_ACCT_PROC_946");
      end;
    end if;

 -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "JET_ISS_ACCT_PROC_946",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end JET_ISS_ACCT_PROC_946_DRIVER;


  procedure JET_RCPT_ACCT_PROC_949_DRIVER is
```

```
    LV_MONTH_JET_RCPT_TOTAL : INTEGER;
    LV_TOTAL_MO_JET_RCPT : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_MONTH_JET_RCPT_TOTAL_JET_RCPT_ACCT_PROC_949.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin

DS_MONTH_JET_RCPT_TOTAL_JET_RCPT_ACCT_PROC_949.BUFFER.READ(LV_MONTH_JET_
RCPT_TOTAL);
    exception
      when BUFFER_UNDERFLOW =>

DS_DEBUG.BUFFER_UNDERFLOW("MONTH_JET_RCPT_TOTAL_JET_RCPT_ACCT_PROC_949"
, "JET_RCPT_ACCT_PROC_949");
    end;

-- Execution trigger condition check.
     if True then
       begin
       JET_RCPT_ACCT_PROC_949(
         MONTH_JET_RCPT_TOTAL => LV_MONTH_JET_RCPT_TOTAL,
         TOTAL_MO_JET_RCPT => LV_TOTAL_MO_JET_RCPT);
       exception
         when others =>
           DS_DEBUG.UNDECLARED_EXCEPTION("JET_RCPT_ACCT_PROC_949");
           EXCEPTION_HAS_OCCURRED := true;
           EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
       end;
     else return;
     end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
     if not EXCEPTION_HAS_OCCURRED then
       begin

DS_TOTAL_MO_JET_RCPT_JET_ACCT_CALC_958.BUFFER.WRITE(LV_TOTAL_MO_JET_RCPT)
;
       exception
         when BUFFER_OVERFLOW =>
           DS_DEBUG.BUFFER_OVERFLOW("TOTAL_MO_JET_RCPT_JET_ACCT_CALC_958",
"JET_RCPT_ACCT_PROC_949");
       end;
     end if;

  -- PSDL Exception handler.
```

```
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "JET_RCPT_ACCT_PROC_949",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end JET_RCPT_ACCT_PROC_949_DRIVER;


  procedure DF_ACCT_CALC_952_DRIVER is
    LV_TOTAL_MO_DF_ISS : INTEGER;
    LV_TOTAL_MO_DF_RCPT : INTEGER;
    LV_DIESEL_QTY_AVAILABLE : INTEGER;
    LV_TOLERANCE_DF : BOOLEAN;
    LV_OPENING_INV_DIESEL : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_TOTAL_MO_DF_ISS_DF_ACCT_CALC_952.NEW_DATA and then
          DS_TOTAL_MO_DF_RCPT_DF_ACCT_CALC_952.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin
      DS_TOTAL_MO_DF_ISS_DF_ACCT_CALC_952.BUFFER.READ(LV_TOTAL_MO_DF_ISS);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("TOTAL_MO_DF_ISS_DF_ACCT_CALC_952",
"DF_ACCT_CALC_952");
    end;
    begin

DS_TOTAL_MO_DF_RCPT_DF_ACCT_CALC_952.BUFFER.READ(LV_TOTAL_MO_DF_RCPT);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("TOTAL_MO_DF_RCPT_DF_ACCT_CALC_952",
"DF_ACCT_CALC_952");
    end;
    begin

DS_OPENING_INV_DIESEL_DF_ACCT_CALC_952.BUFFER.READ(LV_OPENING_INV_DIESEL);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("OPENING_INV_DIESEL_DF_ACCT_CALC_952",
"DF_ACCT_CALC_952");
    end;
    begin

DS_DIESEL_QTY_AVAILABLE_DF_ACCT_CALC_952.BUFFER.READ(LV_DIESEL_QTY_AVAIL
ABLE);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("DIESEL_QTY_AVAILABLE_DF_ACCT_CALC_952",
"DF_ACCT_CALC_952");
```

```
    end;

-- Execution trigger condition check.
    if True then
      begin
      DF_ACCT_CALC_952(
        TOTAL_MO_DF_ISS => LV_TOTAL_MO_DF_ISS,
        TOTAL_MO_DF_RCPT => LV_TOTAL_MO_DF_RCPT,
        DIESEL_QTY_AVAILABLE => LV_DIESEL_QTY_AVAILABLE,
        TOLERANCE_DF => LV_TOLERANCE_DF,
        OPENING_INV_DIESEL => LV_OPENING_INV_DIESEL);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("DF_ACCT_CALC_952");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_TOLERANCE_DF_GUI_ACC_OFFICER_179.BUFFER.WRITE(LV_TOLERANCE_DF);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("TOLERANCE_DF_GUI_ACC_OFFICER_179",
"DF_ACCT_CALC_952");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_OPENING_INV_DIESEL_DF_ACCT_CALC_952.BUFFER.WRITE(LV_OPENING_INV_DIESEL)
;
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("OPENING_INV_DIESEL_DF_ACCT_CALC_952",
"DF_ACCT_CALC_952");
      end;
    end if;

 -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "DF_ACCT_CALC_952",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end DF_ACCT_CALC_952_DRIVER;


  procedure MG_ACCT_CALC_955_DRIVER is
```

```
    LV_TOTAL_MO_MG_ISS : INTEGER;
    LV_TOTAL_MO_MG_RCPT : INTEGER;
    LV_MOGAS_QTY_AVAILABLE : INTEGER;
    LV_TOLERANCE_MG : BOOLEAN;
    LV_OPENING_INV_MOGAS : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
   begin
-- Data trigger checks.
    if not (DS_TOTAL_MO_MG_ISS_MG_ACCT_CALC_955.NEW_DATA and then
         DS_TOTAL_MO_MG_RCPT_MG_ACCT_CALC_955.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin
      DS_TOTAL_MO_MG_ISS_MG_ACCT_CALC_955.BUFFER.READ(LV_TOTAL_MO_MG_ISS);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("TOTAL_MO_MG_ISS_MG_ACCT_CALC_955",
"MG_ACCT_CALC_955");
    end;
    begin

DS_TOTAL_MO_MG_RCPT_MG_ACCT_CALC_955.BUFFER.READ(LV_TOTAL_MO_MG_RCPT);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("TOTAL_MO_MG_RCPT_MG_ACCT_CALC_955",
"MG_ACCT_CALC_955");
    end;
    begin

DS_OPENING_INV_MOGAS_MG_ACCT_CALC_955.BUFFER.READ(LV_OPENING_INV_MOGAS
);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("OPENING_INV_MOGAS_MG_ACCT_CALC_955",
"MG_ACCT_CALC_955");
    end;
    begin

DS_MOGAS_QTY_AVAILABLE_MG_ACCT_CALC_955.BUFFER.READ(LV_MOGAS_QTY_AVAI
LABLE);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("MOGAS_QTY_AVAILABLE_MG_ACCT_CALC_955",
"MG_ACCT_CALC_955");
    end;

-- Execution trigger condition check.
     if True then
      begin
      MG_ACCT_CALC_955(
        TOTAL_MO_MG_ISS => LV_TOTAL_MO_MG_ISS,
        TOTAL_MO_MG_RCPT => LV_TOTAL_MO_MG_RCPT,
```

```
      MOGAS_QTY_AVAILABLE => LV_MOGAS_QTY_AVAILABLE,
      TOLERANCE_MG => LV_TOLERANCE_MG,
      OPENING_INV_MOGAS => LV_OPENING_INV_MOGAS);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("MG_ACCT_CALC_955");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
   else return;
   end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
   if not EXCEPTION_HAS_OCCURRED then
     begin
       DS_TOLERANCE_MG_GUI_ACC_OFFICER_179.BUFFER.WRITE(LV_TOLERANCE_MG);
     exception
       when BUFFER_OVERFLOW =>
         DS_DEBUG.BUFFER_OVERFLOW("TOLERANCE_MG_GUI_ACC_OFFICER_179",
"MG_ACCT_CALC_955");
     end;
   end if;
   if not EXCEPTION_HAS_OCCURRED then
     begin

DS_OPENING_INV_MOGAS_MG_ACCT_CALC_955.BUFFER.WRITE(LV_OPENING_INV_MOGA
S);
     exception
       when BUFFER_OVERFLOW =>
         DS_DEBUG.BUFFER_OVERFLOW("OPENING_INV_MOGAS_MG_ACCT_CALC_955",
"MG_ACCT_CALC_955");
     end;
   end if;

 -- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
     DS_DEBUG.UNHANDLED_EXCEPTION(
       "MG_ACCT_CALC_955",
       PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
   end if;
  end MG_ACCT_CALC_955_DRIVER;


  procedure JET_ACCT_CALC_958_DRIVER is
   LV_TOTAL_MO_JET_ISS : INTEGER;
   LV_TOTAL_MO_JET_RCPT : INTEGER;
   LV_JET_QTY_AVAILABLE : INTEGER;
   LV_TOLERANCE_JET : BOOLEAN;
   LV_OPENING_INV_JET : INTEGER;

   EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
   EXCEPTION_ID: PSDL_EXCEPTION;
```

```
   begin
-- Data trigger checks.
    if not (DS_TOTAL_MO_JET_ISS_JET_ACCT_CALC_958.NEW_DATA and then
          DS_TOTAL_MO_JET_RCPT_JET_ACCT_CALC_958.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin
      DS_TOTAL_MO_JET_ISS_JET_ACCT_CALC_958.BUFFER.READ(LV_TOTAL_MO_JET_ISS);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("TOTAL_MO_JET_ISS_JET_ACCT_CALC_958",
"JET_ACCT_CALC_958");
    end;
    begin

DS_TOTAL_MO_JET_RCPT_JET_ACCT_CALC_958.BUFFER.READ(LV_TOTAL_MO_JET_RCPT);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("TOTAL_MO_JET_RCPT_JET_ACCT_CALC_958",
"JET_ACCT_CALC_958");
    end;
    begin
      DS_OPENING_INV_JET_JET_ACCT_CALC_958.BUFFER.READ(LV_OPENING_INV_JET);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("OPENING_INV_JET_JET_ACCT_CALC_958",
"JET_ACCT_CALC_958");
    end;
    begin

DS_JET_QTY_AVAILABLE_JET_ACCT_CALC_958.BUFFER.READ(LV_JET_QTY_AVAILABLE);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("JET_QTY_AVAILABLE_JET_ACCT_CALC_958",
"JET_ACCT_CALC_958");
    end;

-- Execution trigger condition check.
    if True then
      begin
      JET_ACCT_CALC_958(
        TOTAL_MO_JET_ISS => LV_TOTAL_MO_JET_ISS,
        TOTAL_MO_JET_RCPT => LV_TOTAL_MO_JET_RCPT,
        JET_QTY_AVAILABLE => LV_JET_QTY_AVAILABLE,
        TOLERANCE_JET => LV_TOLERANCE_JET,
        OPENING_INV_JET => LV_OPENING_INV_JET);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("JET_ACCT_CALC_958");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;
```

```
-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_TOLERANCE_JET_GUI_ACC_OFFICER_179.BUFFER.WRITE(LV_TOLERANCE_JET);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("TOLERANCE_JET_GUI_ACC_OFFICER_179",
"JET_ACCT_CALC_958");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_OPENING_INV_JET_JET_ACCT_CALC_958.BUFFER.WRITE(LV_OPENING_INV_JET);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("OPENING_INV_JET_JET_ACCT_CALC_958",
"JET_ACCT_CALC_958");
      end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "JET_ACCT_CALC_958",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end JET_ACCT_CALC_958_DRIVER;


  procedure DIESEL_GAGE_854_DRIVER is
    LV_DF_QTY_ON_HAND : INTEGER;
    LV_DIESEL_QTY_AVAILABLE : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
-- Data trigger checks.
    if not (DS_DF_QTY_ON_HAND_DIESEL_GAGE_854.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin
      DS_DF_QTY_ON_HAND_DIESEL_GAGE_854.BUFFER.READ(LV_DF_QTY_ON_HAND);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("DF_QTY_ON_HAND_DIESEL_GAGE_854",
"DIESEL_GAGE_854");
    end;

-- Execution trigger condition check.
```

```
    if True then
      begin
      DIESEL_GAGE_854(
        DF_QTY_ON_HAND => LV_DF_QTY_ON_HAND,
        DIESEL_QTY_AVAILABLE => LV_DIESEL_QTY_AVAILABLE);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("DIESEL_GAGE_854");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin

DS_DIESEL_QTY_AVAILABLE_DF_ACCT_CALC_952.BUFFER.WRITE(LV_DIESEL_QTY_AVAI
LABLE);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("DIESEL_QTY_AVAILABLE_DF_ACCT_CALC_952",
"DIESEL_GAGE_854");
      end;
      begin

DS_DIESEL_QTY_AVAILABLE_GUI_FUEL_ON_HAND_124.BUFFER.WRITE(LV_DIESEL_QTY_
AVAILABLE);
      exception
        when BUFFER_OVERFLOW =>

DS_DEBUG.BUFFER_OVERFLOW("DIESEL_QTY_AVAILABLE_GUI_FUEL_ON_HAND_124",
"DIESEL_GAGE_854");
      end;
    end if;

-- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "DIESEL_GAGE_854",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end DIESEL_GAGE_854_DRIVER;


  procedure DIESEL_SUBTRACTION_839_DRIVER is
    LV_DIESEL_ISS_QTY : INTEGER;
    LV_DF_QTY_ON_HAND : INTEGER;
    LV_DIESEL_VOLUME : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
```

```
    EXCEPTION_ID: PSDL_EXCEPTION;
   begin
-- Data trigger checks.
    if not (DS_DIESEL_ISS_QTY_DIESEL_SUBTRACTION_839.NEW_DATA) then
      return;
    end if;

-- Data stream reads.
    begin

DS_DIESEL_VOLUME_DIESEL_SUBTRACTION_839.BUFFER.READ(LV_DIESEL_VOLUME);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("DIESEL_VOLUME_DIESEL_SUBTRACTION_839",
"DIESEL_SUBTRACTION_839");
    end;
    begin
      DS_DIESEL_ISS_QTY_DIESEL_SUBTRACTION_839.BUFFER.READ(LV_DIESEL_ISS_QTY);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("DIESEL_ISS_QTY_DIESEL_SUBTRACTION_839",
"DIESEL_SUBTRACTION_839");
    end;

-- Execution trigger condition check.
    if True then
      begin
      DIESEL_SUBTRACTION_839(
        DIESEL_ISS_QTY => LV_DIESEL_ISS_QTY,
        DF_QTY_ON_HAND => LV_DF_QTY_ON_HAND,
        DIESEL_VOLUME => LV_DIESEL_VOLUME);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("DIESEL_SUBTRACTION_839");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 --Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_DF_QTY_ON_HAND_DIESEL_GAGE_854.BUFFER.WRITE(LV_DF_QTY_ON_HAND);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("DF_QTY_ON_HAND_DIESEL_GAGE_854",
 "DIESEL_SUBTRACTION_839");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin
```

```
      DS_DIESEL_VOLUME_DIESEL_ADDITION_836.BUFFER.WRITE(LV_DIESEL_VOLUME);
     exception
      when BUFFER_OVERFLOW =>
       DS_DEBUG.BUFFER_OVERFLOW("DIESEL_VOLUME_DIESEL_ADDITION_836",
"DIESEL_SUBTRACTION_839");
     end;
     begin

DS_DIESEL_VOLUME_DIESEL_SUBTRACTION_839.BUFFER.WRITE(LV_DIESEL_VOLUME);
     exception
      when BUFFER_OVERFLOW =>
       DS_DEBUG.BUFFER_OVERFLOW("DIESEL_VOLUME_DIESEL_SUBTRACTION_839",
"DIESEL_SUBTRACTION_839");
     end;
    end if;

 -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
     DS_DEBUG.UNHANDLED_EXCEPTION(
      "DIESEL_SUBTRACTION_839",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
   end DIESEL_SUBTRACTION_839_DRIVER;


   procedure DIESEL_ADDITION_836_DRIVER is
    LV_DIESEL_RCPT_QTY : INTEGER;
    LV_DF_QTY_ON_HAND : INTEGER;
    LV_DIESEL_VOLUME : INTEGER;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
   begin
-- Data trigger checks.
    if not (DS_DIESEL_RCPT_QTY_DIESEL_ADDITION_836.NEW_DATA) then
     return;
    end if;

-- Data stream reads.
    begin
     DS_DIESEL_VOLUME_DIESEL_ADDITION_836.BUFFER.READ(LV_DIESEL_VOLUME);
    exception
     when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("DIESEL_VOLUME_DIESEL_ADDITION_836",
"DIESEL_ADDITION_836");
    end;
    begin
     DS_DIESEL_RCPT_QTY_DIESEL_ADDITION_836.BUFFER.READ(LV_DIESEL_RCPT_QTY);
    exception
     when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("DIESEL_RCPT_QTY_DIESEL_ADDITION_836",
"DIESEL_ADDITION_836");
    end;

-- Execution trigger condition check.
    if True then
```

```
      begin
      DIESEL_ADDITION_836(
        DIESEL_RCPT_QTY => LV_DIESEL_RCPT_QTY,
        DF_QTY_ON_HAND => LV_DF_QTY_ON_HAND,
        DIESEL_VOLUME => LV_DIESEL_VOLUME);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("DIESEL_ADDITION_836");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

-- Exception Constraint translations.

-- Other constraint option translations.

--Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_DF_QTY_ON_HAND_DIESEL_GAGE_854.BUFFER.WRITE(LV_DF_QTY_ON_HAND);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("DF_QTY_ON_HAND_DIESEL_GAGE_854",
"DIESEL_ADDITION_836");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_DIESEL_VOLUME_DIESEL_ADDITION_836.BUFFER.WRITE(LV_DIESEL_VOLUME);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("DIESEL_VOLUME_DIESEL_ADDITION_836",
"DIESEL_ADDITION_836");
      end;
      begin

DS_DIESEL_VOLUME_DIESEL_SUBTRACTION_839.BUFFER.WRITE(LV_DIESEL_VOLUME);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("DIESEL_VOLUME_DIESEL_SUBTRACTION_839",
"DIESEL_ADDITION_836");
      end;
    end if;

 -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "DIESEL_ADDITION_836",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end DIESEL_ADDITION_836_DRIVER;
 end FUEL_SUBSYSTEM_1_DRIVERS;

 package fuel_subsystem_1_DYNAMIC_SCHEDULERS is
```

```
  procedure START_DYNAMIC_SCHEDULE;
  procedure STOP_DYNAMIC_SCHEDULE;
end fuel_subsystem_1_DYNAMIC_SCHEDULERS;


with fuel_subsystem_1_DRIVERS;  use fuel_subsystem_1_DRIVERS;
with PRIORITY_DEFINITIONS;  use PRIORITY_DEFINITIONS;
package body fuel_subsystem_1_DYNAMIC_SCHEDULERS is

  task type DYNAMIC_SCHEDULE_TYPE is
    pragma priority (DYNAMIC_SCHEDULE_PRIORITY);
    entry START;
  end DYNAMIC_SCHEDULE_TYPE;
  for DYNAMIC_SCHEDULE_TYPE'STORAGE_SIZE use 100_000;
  DYNAMIC_SCHEDULE : DYNAMIC_SCHEDULE_TYPE;

  done : boolean := false;
  procedure STOP_DYNAMIC_SCHEDULE is
  begin
    done := true;
  end STOP_DYNAMIC_SCHEDULE;

  task body DYNAMIC_SCHEDULE_TYPE is
  begin
    accept START;
    loop
      gui_bulk_receipt_3_DRIVER;
      exit when done;

      gui_other_receipt_6_DRIVER;
      exit when done;

      gui_bulk_issue_9_DRIVER;
      exit when done;

      gui_other_issue_12_DRIVER;
      exit when done;

      jet_gage_917_DRIVER;
      exit when done;

      mogas_gage_894_DRIVER;
      exit when done;

      diesel_gage_854_DRIVER;
      exit when done;

      bulk_rcpt_db_table_495_DRIVER;
      exit when done;

      bulk_rcpt_processor_198_DRIVER;
      exit when done;

      other_rcpt_db_table_501_DRIVER;
      exit when done;
```

```
oth_rcpt_processor_207_DRIVER;
exit when done;

rcpt_processor_210_DRIVER;
exit when done;

bulk_iss_db_table_498_DRIVER;
exit when done;

bulk_iss_processor_310_DRIVER;
exit when done;

other_iss_db_table_504_DRIVER;
exit when done;

oth_iss_processor_307_DRIVER;
exit when done;

iss_processor_323_DRIVER;
exit when done;

jet_acct_calc_958_DRIVER;
exit when done;

mg_acct_calc_955_DRIVER;
exit when done;

daily_rcpt_db_table_740_DRIVER;
exit when done;

mo_df_rcpt_totalizer_583_DRIVER;
exit when done;

daily_iss_db_table_743_DRIVER;
exit when done;

mo_df_iss_totalizer_592_DRIVER;
exit when done;

mo_jet_iss_totalizer_598_DRIVER;
exit when done;

mo_jet_rcpt_totalizer_589_DRIVER;
exit when done;

mo_mg_iss_totalizer_595_DRIVER;
exit when done;

mo_mg_rcpt_totalizer_586_DRIVER;
exit when done;

diesel_iss_acct_proc_934_DRIVER;
exit when done;

diesel_rcpt_acct_proc_937_DRIVER;
exit when done;
```

```
jet_iss_acct_proc_946_DRIVER;
exit when done;

jet_rcpt_acct_proc_949_DRIVER;
exit when done;

mogas_iss_acct_proc_940_DRIVER;
exit when done;

mogas_rcpt_acct_proc_943_DRIVER;
exit when done;

gui_fuel_on_hand_124_DRIVER;
exit when done;

df_acct_calc_952_DRIVER;
exit when done;

jet_rcpt_totalizer_280_DRIVER;
exit when done;

mg_rcpt_totalizer_277_DRIVER;
exit when done;

df_rcpt_totalizer_274_DRIVER;
exit when done;

jet_addition_911_DRIVER;
exit when done;

mogas_addition_888_DRIVER;
exit when done;

diesel_addition_836_DRIVER;
exit when done;

df_iss_totalizer_352_DRIVER;
exit when done;

mg_iss_totalizer_349_DRIVER;
exit when done;

jet_iss_totalizer_346_DRIVER;
exit when done;

jet_subtraction_914_DRIVER;
exit when done;

mogas_subtraction_891_DRIVER;
exit when done;

diesel_subtraction_839_DRIVER;
exit when done;

gui_acc_officer_179_DRIVER;
```

```
    exit when done;

  end loop;
 end DYNAMIC_SCHEDULE_TYPE;

 procedure START_DYNAMIC_SCHEDULE is
 begin
   DYNAMIC_SCHEDULE.START;
 end START_DYNAMIC_SCHEDULE;

end fuel_subsystem_1_DYNAMIC_SCHEDULERS;


package fuel_subsystem_1_STATIC_SCHEDULERS is
 procedure START_STATIC_SCHEDULE;
 procedure STOP_STATIC_SCHEDULE;
end fuel_subsystem_1_STATIC_SCHEDULERS;

with fuel_subsystem_1_DRIVERS;  use fuel_subsystem_1_DRIVERS;
with PRIORITY_DEFINITIONS;  use PRIORITY_DEFINITIONS;
with PSDL_TIMERS;  use PSDL_TIMERS;
with TEXT_IO;  use TEXT_IO;
package body fuel_subsystem_1_STATIC_SCHEDULERS is

 task type STATIC_SCHEDULE_TYPE is
   pragma priority (STATIC_SCHEDULE_PRIORITY);
   entry START;
 end STATIC_SCHEDULE_TYPE;
 for STATIC_SCHEDULE_TYPE'STORAGE_SIZE use 200_000;
 STATIC_SCHEDULE : STATIC_SCHEDULE_TYPE;

 done : boolean := false;
 procedure STOP_STATIC_SCHEDULE is
 begin
   done := true;
 end STOP_STATIC_SCHEDULE;

 task body STATIC_SCHEDULE_TYPE is
   PERIOD : duration;
   daily_reporter_410_START_TIME1 : duration;
   daily_reporter_410_STOP_TIME1 : duration;
   monthly_reporter_601_START_TIME2 : duration;
   monthly_reporter_601_STOP_TIME2 : duration;
   daily_reporter_410_START_TIME3 : duration;
   daily_reporter_410_STOP_TIME3 : duration;
   daily_reporter_410_START_TIME4 : duration;
   daily_reporter_410_STOP_TIME4 : duration;
   daily_reporter_410_START_TIME5 : duration;
   daily_reporter_410_STOP_TIME5 : duration;
   daily_reporter_410_START_TIME6 : duration;
   daily_reporter_410_STOP_TIME6 : duration;
   daily_reporter_410_START_TIME7 : duration;
   daily_reporter_410_STOP_TIME7 : duration;
   daily_reporter_410_START_TIME8 : duration;
   daily_reporter_410_STOP_TIME8 : duration;
   daily_reporter_410_START_TIME9 : duration;
```

```
  daily_reporter_410_STOP_TIME9 : duration;
  daily_reporter_410_START_TIME10 : duration;
  daily_reporter_410_STOP_TIME10 : duration;
  daily_reporter_410_START_TIME11 : duration;
  daily_reporter_410_STOP_TIME11 : duration;
  daily_reporter_410_START_TIME12 : duration;
  daily_reporter_410_STOP_TIME12 : duration;
  daily_reporter_410_START_TIME13 : duration;
  daily_reporter_410_STOP_TIME13 : duration;
  daily_reporter_410_START_TIME14 : duration;
  daily_reporter_410_STOP_TIME14 : duration;
  daily_reporter_410_START_TIME15 : duration;
  daily_reporter_410_STOP_TIME15 : duration;
  daily_reporter_410_START_TIME16 : duration;
  daily_reporter_410_STOP_TIME16 : duration;
  daily_reporter_410_START_TIME17 : duration;
  daily_reporter_410_STOP_TIME17 : duration;
  daily_reporter_410_START_TIME18 : duration;
  daily_reporter_410_STOP_TIME18 : duration;
  daily_reporter_410_START_TIME19 : duration;
  daily_reporter_410_STOP_TIME19 : duration;
  daily_reporter_410_START_TIME20 : duration;
  daily_reporter_410_STOP_TIME20 : duration;
  daily_reporter_410_START_TIME21 : duration;
  daily_reporter_410_STOP_TIME21 : duration;
  daily_reporter_410_START_TIME22 : duration;
  daily_reporter_410_STOP_TIME22 : duration;
  daily_reporter_410_START_TIME23 : duration;
  daily_reporter_410_STOP_TIME23 : duration;
  daily_reporter_410_START_TIME24 : duration;
  daily_reporter_410_STOP_TIME24 : duration;
  daily_reporter_410_START_TIME25 : duration;
  daily_reporter_410_STOP_TIME25 : duration;
  daily_reporter_410_START_TIME26 : duration;
  daily_reporter_410_STOP_TIME26 : duration;
  daily_reporter_410_START_TIME27 : duration;
  daily_reporter_410_STOP_TIME27 : duration;
  daily_reporter_410_START_TIME28 : duration;
  daily_reporter_410_STOP_TIME28 : duration;
  daily_reporter_410_START_TIME29 : duration;
  daily_reporter_410_STOP_TIME29 : duration;
  daily_reporter_410_START_TIME30 : duration;
  daily_reporter_410_STOP_TIME30 : duration;
  daily_reporter_410_START_TIME31 : duration;
  daily_reporter_410_STOP_TIME31 : duration;
  schedule_timer : TIMER := NEW_TIMER;
begin
  accept START;
  PERIOD := TARGET_TO_HOST(duration( 1.08000E+05));
  daily_reporter_410_START_TIME1 := TARGET_TO_HOST(duration( 0.00000E+00));
  daily_reporter_410_STOP_TIME1 := TARGET_TO_HOST(duration( 7.50000E-01));
  monthly_reporter_601_START_TIME2 := TARGET_TO_HOST(duration( 7.50000E-01));
  monthly_reporter_601_STOP_TIME2 := TARGET_TO_HOST(duration( 1.50000E+00));
  daily_reporter_410_START_TIME3 := TARGET_TO_HOST(duration( 3.60000E+03));
  daily_reporter_410_STOP_TIME3 := TARGET_TO_HOST(duration( 3.60075E+03));
  daily_reporter_410_START_TIME4 := TARGET_TO_HOST(duration( 7.20000E+03));
```

```
daily_reporter_410_STOP_TIME4 := TARGET_TO_HOST(duration( 7.20075E+03));
daily_reporter_410_START_TIME5 := TARGET_TO_HOST(duration( 1.08000E+04));
daily_reporter_410_STOP_TIME5 := TARGET_TO_HOST(duration( 1.08008E+04));
daily_reporter_410_START_TIME6 := TARGET_TO_HOST(duration( 1.44000E+04));
daily_reporter_410_STOP_TIME6 := TARGET_TO_HOST(duration( 1.44008E+04));
daily_reporter_410_START_TIME7 := TARGET_TO_HOST(duration( 1.80000E+04));
daily_reporter_410_STOP_TIME7 := TARGET_TO_HOST(duration( 1.80008E+04));
daily_reporter_410_START_TIME8 := TARGET_TO_HOST(duration( 2.16000E+04));
daily_reporter_410_STOP_TIME8 := TARGET_TO_HOST(duration( 2.16008E+04));
daily_reporter_410_START_TIME9 := TARGET_TO_HOST(duration( 2.52000E+04));
daily_reporter_410_STOP_TIME9 := TARGET_TO_HOST(duration( 2.52008E+04));
daily_reporter_410_START_TIME10 := TARGET_TO_HOST(duration( 2.88000E+04));
daily_reporter_410_STOP_TIME10 := TARGET_TO_HOST(duration( 2.88008E+04));
daily_reporter_410_START_TIME11 := TARGET_TO_HOST(duration( 3.24000E+04));
daily_reporter_410_STOP_TIME11 := TARGET_TO_HOST(duration( 3.24008E+04));
daily_reporter_410_START_TIME12 := TARGET_TO_HOST(duration( 3.60000E+04));
daily_reporter_410_STOP_TIME12 := TARGET_TO_HOST(duration( 3.60008E+04));
daily_reporter_410_START_TIME13 := TARGET_TO_HOST(duration( 3.96000E+04));
daily_reporter_410_STOP_TIME13 := TARGET_TO_HOST(duration( 3.96008E+04));
daily_reporter_410_START_TIME14 := TARGET_TO_HOST(duration( 4.32000E+04));
daily_reporter_410_STOP_TIME14 := TARGET_TO_HOST(duration( 4.32008E+04));
daily_reporter_410_START_TIME15 := TARGET_TO_HOST(duration( 4.68000E+04));
daily_reporter_410_STOP_TIME15 := TARGET_TO_HOST(duration( 4.68008E+04));
daily_reporter_410_START_TIME16 := TARGET_TO_HOST(duration( 5.04000E+04));
daily_reporter_410_STOP_TIME16 := TARGET_TO_HOST(duration( 5.04008E+04));
daily_reporter_410_START_TIME17 := TARGET_TO_HOST(duration( 5.40000E+04));
daily_reporter_410_STOP_TIME17 := TARGET_TO_HOST(duration( 5.40008E+04));
daily_reporter_410_START_TIME18 := TARGET_TO_HOST(duration( 5.76000E+04));
daily_reporter_410_STOP_TIME18 := TARGET_TO_HOST(duration( 5.76008E+04));
daily_reporter_410_START_TIME19 := TARGET_TO_HOST(duration( 6.12000E+04));
daily_reporter_410_STOP_TIME19 := TARGET_TO_HOST(duration( 6.12008E+04));
daily_reporter_410_START_TIME20 := TARGET_TO_HOST(duration( 6.48000E+04));
daily_reporter_410_STOP_TIME20 := TARGET_TO_HOST(duration( 6.48008E+04));
daily_reporter_410_START_TIME21 := TARGET_TO_HOST(duration( 6.84000E+04));
daily_reporter_410_STOP_TIME21 := TARGET_TO_HOST(duration( 6.84008E+04));
daily_reporter_410_START_TIME22 := TARGET_TO_HOST(duration( 7.20000E+04));
daily_reporter_410_STOP_TIME22 := TARGET_TO_HOST(duration( 7.20008E+04));
daily_reporter_410_START_TIME23 := TARGET_TO_HOST(duration( 7.56000E+04));
daily_reporter_410_STOP_TIME23 := TARGET_TO_HOST(duration( 7.56008E+04));
daily_reporter_410_START_TIME24 := TARGET_TO_HOST(duration( 7.92000E+04));
daily_reporter_410_STOP_TIME24 := TARGET_TO_HOST(duration( 7.92008E+04));
daily_reporter_410_START_TIME25 := TARGET_TO_HOST(duration( 8.28000E+04));
daily_reporter_410_STOP_TIME25 := TARGET_TO_HOST(duration( 8.28008E+04));
daily_reporter_410_START_TIME26 := TARGET_TO_HOST(duration( 8.64000E+04));
daily_reporter_410_STOP_TIME26 := TARGET_TO_HOST(duration( 8.64008E+04));
daily_reporter_410_START_TIME27 := TARGET_TO_HOST(duration( 9.00000E+04));
daily_reporter_410_STOP_TIME27 := TARGET_TO_HOST(duration( 9.00008E+04));
daily_reporter_410_START_TIME28 := TARGET_TO_HOST(duration( 9.36000E+04));
daily_reporter_410_STOP_TIME28 := TARGET_TO_HOST(duration( 9.36008E+04));
daily_reporter_410_START_TIME29 := TARGET_TO_HOST(duration( 9.72000E+04));
daily_reporter_410_STOP_TIME29 := TARGET_TO_HOST(duration( 9.72008E+04));
daily_reporter_410_START_TIME30 := TARGET_TO_HOST(duration( 1.00800E+05));
daily_reporter_410_STOP_TIME30 := TARGET_TO_HOST(duration( 1.00801E+05));
daily_reporter_410_START_TIME31 := TARGET_TO_HOST(duration( 1.04400E+05));
daily_reporter_410_STOP_TIME31 := TARGET_TO_HOST(duration( 1.04401E+05));
START(schedule_timer);
```

```
  loop
    delay(daily_reporter_410_START_TIME1 - HOST_DURATION(schedule_timer));
    daily_reporter_410_DRIVER;
    if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME1 then
      PUT_LINE("timing error from operator daily_reporter_410");
      SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME1);
    end if;
    exit when done;

    delay(monthly_reporter_601_START_TIME2 - HOST_DURATION(schedule_timer));
    monthly_reporter_601_DRIVER;
    if HOST_DURATION(schedule_timer) > monthly_reporter_601_STOP_TIME2 then
      PUT_LINE("timing error from operator monthly_reporter_601");
      SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
monthly_reporter_601_STOP_TIME2);
    end if;
    exit when done;

    delay(daily_reporter_410_START_TIME3 - HOST_DURATION(schedule_timer));
    daily_reporter_410_DRIVER;
    if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME3 then
      PUT_LINE("timing error from operator daily_reporter_410");
      SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME3);
    end if;
    exit when done;

    delay(daily_reporter_410_START_TIME4 - HOST_DURATION(schedule_timer));
    daily_reporter_410_DRIVER;
    if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME4 then
      PUT_LINE("timing error from operator daily_reporter_410");
      SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME4);
    end if;
    exit when done;

    delay(daily_reporter_410_START_TIME5 - HOST_DURATION(schedule_timer));
    daily_reporter_410_DRIVER;
    if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME5 then
      PUT_LINE("timing error from operator daily_reporter_410");
      SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME5);
    end if;
    exit when done;

    delay(daily_reporter_410_START_TIME6 - HOST_DURATION(schedule_timer));
    daily_reporter_410_DRIVER;
    if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME6 then
      PUT_LINE("timing error from operator daily_reporter_410");
      SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME6);
    end if;
    exit when done;

    delay(daily_reporter_410_START_TIME7 - HOST_DURATION(schedule_timer));
```

```
      daily_reporter_410_DRIVER;
      if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME7 then
        PUT_LINE("timing error from operator daily_reporter_410");
        SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME7);
      end if;
      exit when done;

      delay(daily_reporter_410_START_TIME8 - HOST_DURATION(schedule_timer));
      daily_reporter_410_DRIVER;
      if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME8 then
        PUT_LINE("timing error from operator daily_reporter_410");
        SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME8);
      end if;
      exit when done;

      delay(daily_reporter_410_START_TIME9 - HOST_DURATION(schedule_timer));
      daily_reporter_410_DRIVER;
      if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME9 then
        PUT_LINE("timing error from operator daily_reporter_410");
        SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME9);
      end if;
      exit when done;

      delay(daily_reporter_410_START_TIME10 - HOST_DURATION(schedule_timer));
      daily_reporter_410_DRIVER;
      if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME10 then
        PUT_LINE("timing error from operator daily_reporter_410");
        SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME10);
      end if;
      exit when done;

      delay(daily_reporter_410_START_TIME11 - HOST_DURATION(schedule_timer));
      daily_reporter_410_DRIVER;
      if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME11 then
        PUT_LINE("timing error from operator daily_reporter_410");
        SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME11);
      end if;
      exit when done;

      delay(daily_reporter_410_START_TIME12 - HOST_DURATION(schedule_timer));
      daily_reporter_410_DRIVER;
      if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME12 then
        PUT_LINE("timing error from operator daily_reporter_410");
        SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME12);
      end if;
      exit when done;

      delay(daily_reporter_410_START_TIME13 - HOST_DURATION(schedule_timer));
      daily_reporter_410_DRIVER;
      if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME13 then
```

```
    PUT_LINE("timing error from operator daily_reporter_410");
    SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME13);
  end if;
  exit when done;

  delay(daily_reporter_410_START_TIME14 - HOST_DURATION(schedule_timer));
  daily_reporter_410_DRIVER;
  if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME14 then
    PUT_LINE("timing error from operator daily_reporter_410");
    SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME14);
  end if;
  exit when done;

  delay(daily_reporter_410_START_TIME15 - HOST_DURATION(schedule_timer));
  daily_reporter_410_DRIVER;
  if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME15 then
    PUT_LINE("timing error from operator daily_reporter_410");
    SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME15);
  end if;
  exit when done;

  delay(daily_reporter_410_START_TIME16 - HOST_DURATION(schedule_timer));
  daily_reporter_410_DRIVER;
  if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME16 then
    PUT_LINE("timing error from operator daily_reporter_410");
    SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME16);
  end if;
  exit when done;

  delay(daily_reporter_410_START_TIME17 - HOST_DURATION(schedule_timer));
  daily_reporter_410_DRIVER;
  if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME17 then
    PUT_LINE("timing error from operator daily_reporter_410");
    SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME17);
  end if;
  exit when done;

  delay(daily_reporter_410_START_TIME18 - HOST_DURATION(schedule_timer));
  daily_reporter_410_DRIVER;
  if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME18 then
    PUT_LINE("timing error from operator daily_reporter_410");
    SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME18);
  end if;
  exit when done;

  delay(daily_reporter_410_START_TIME19 - HOST_DURATION(schedule_timer));
  daily_reporter_410_DRIVER;
  if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME19 then
    PUT_LINE("timing error from operator daily_reporter_410");
```

```
    SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME19);
   end if;
   exit when done;

   delay(daily_reporter_410_START_TIME20 - HOST_DURATION(schedule_timer));
   daily_reporter_410_DRIVER;
   if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME20 then
    PUT_LINE("timing error from operator daily_reporter_410");
    SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME20);
   end if;
   exit when done;

   delay(daily_reporter_410_START_TIME21 - HOST_DURATION(schedule_timer));
   daily_reporter_410_DRIVER;
   if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME21 then
    PUT_LINE("timing error from operator daily_reporter_410");
    SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME21);
   end if;
   exit when done;

   delay(daily_reporter_410_START_TIME22 - HOST_DURATION(schedule_timer));
   daily_reporter_410_DRIVER;
   if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME22 then
    PUT_LINE("timing error from operator daily_reporter_410");
    SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME22);
   end if;
   exit when done;

   delay(daily_reporter_410_START_TIME23 - HOST_DURATION(schedule_timer));
   daily_reporter_410_DRIVER;
   if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME23 then
    PUT_LINE("timing error from operator daily_reporter_410");
    SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME23);
   end if;
   exit when done;

   delay(daily_reporter_410_START_TIME24 - HOST_DURATION(schedule_timer));
   daily_reporter_410_DRIVER;
   if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME24 then
    PUT_LINE("timing error from operator daily_reporter_410");
    SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME24);
   end if;
   exit when done;

   delay(daily_reporter_410_START_TIME25 - HOST_DURATION(schedule_timer));
   daily_reporter_410_DRIVER;
   if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME25 then
    PUT_LINE("timing error from operator daily_reporter_410");
    SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME25);
```

```
    end if;
    exit when done;

    delay(daily_reporter_410_START_TIME26 - HOST_DURATION(schedule_timer));
    daily_reporter_410_DRIVER;
    if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME26 then
      PUT_LINE("timing error from operator daily_reporter_410");
      SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME26);
    end if;
    exit when done;

    delay(daily_reporter_410_START_TIME27 - HOST_DURATION(schedule_timer));
    daily_reporter_410_DRIVER;
    if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME27 then
      PUT_LINE("timing error from operator daily_reporter_410");
      SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME27);
    end if;
    exit when done;

    delay(daily_reporter_410_START_TIME28 - HOST_DURATION(schedule_timer));
    daily_reporter_410_DRIVER;
    if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME28 then
      PUT_LINE("timing error from operator daily_reporter_410");
      SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME28);
    end if;
    exit when done;

    delay(daily_reporter_410_START_TIME29 - HOST_DURATION(schedule_timer));
    daily_reporter_410_DRIVER;
    if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME29 then
      PUT_LINE("timing error from operator daily_reporter_410");
      SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME29);
    end if;
    exit when done;

    delay(daily_reporter_410_START_TIME30 - HOST_DURATION(schedule_timer));
    daily_reporter_410_DRIVER;
    if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME30 then
      PUT_LINE("timing error from operator daily_reporter_410");
      SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME30);
    end if;
    exit when done;

    delay(daily_reporter_410_START_TIME31 - HOST_DURATION(schedule_timer));
    daily_reporter_410_DRIVER;
    if HOST_DURATION(schedule_timer) > daily_reporter_410_STOP_TIME31 then
      PUT_LINE("timing error from operator daily_reporter_410");
      SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
daily_reporter_410_STOP_TIME31);
    end if;
    exit when done;
```

```
      delay(PERIOD - HOST_DURATION(schedule_timer));
      RESET(schedule_timer);
    end loop;
  end STATIC_SCHEDULE_TYPE;

  procedure START_STATIC_SCHEDULE is
  begin
    STATIC_SCHEDULE.START;
  end START_STATIC_SCHEDULE;

end fuel_subsystem_1_STATIC_SCHEDULERS;


  with FUEL_SUBSYSTEM_1_STATIC_SCHEDULERS; use
FUEL_SUBSYSTEM_1_STATIC_SCHEDULERS;
  with FUEL_SUBSYSTEM_1_DYNAMIC_SCHEDULERS; use
FUEL_SUBSYSTEM_1_DYNAMIC_SCHEDULERS;
  with CAPS_HARDWARE_MODEL; use CAPS_HARDWARE_MODEL;
 procedure FUEL_SUBSYSTEM_1 is
 begin
  init_hardware_model;
  start_static_schedule;
  start_dynamic_schedule;
 end FUEL_SUBSYSTEM_1;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.gui_other_issue_12.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  this package simulates the interface that allows a petroleum
--            specialist to manually input parameters required by Army
--            Regulation 710-2 for all fuel issues other than bulk
--            issues.  These other issues include fuel issues made directly
--            into or specifically identifiable to a consuming end item.
--            An example of this type of issue is to a vehicle or a M2
--            burner unit.  The petroleum specialist enters the type of
--            fuel issued, quantity in gallons, the receiving vehicle
--            bumper number/equipment name, the receiving unit, and the
--            name/rank of the receiver.
--
--------------------------------------------------------------------------------


with text_string_pkg; use text_string_pkg;

package gui_other_issue_12_pkg is

  procedure gui_other_issue_12(
    eq_iss_unit: out text_string;
    eq_iss_name: out text_string;
    eq_iss_fuel_type: out integer;
    eq_iss_id: out text_string;
    eq_iss_qty: out integer );
end gui_other_issue_12_pkg;

package body gui_other_issue_12_pkg is

  procedure gui_other_issue_12(
    eq_iss_unit: out text_string;
    eq_iss_name: out text_string;
    eq_iss_fuel_type: out integer;
    eq_iss_id: out text_string;
    eq_iss_qty: out integer ) is

  begin

    null;  -- the interface would be implemented here

  end gui_other_issue_12;

end gui_other_issue_12_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.gui_fuel_on_hand_124.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package simulates the interface that displays the
--              current fuel totals in gallons that are stored and available
--              for issue.  This interface is available to both the using
--              petroleum specialist and accountable officer.
--
--------------------------------------------------------------------------------


package gui_fuel_on_hand_124_pkg is

  procedure gui_fuel_on_hand_124(
    diesel_qty_available: in integer;
    mogas_qty_available: in integer;
    jet_qty_available: in integer );
end gui_fuel_on_hand_124_pkg;

package body gui_fuel_on_hand_124_pkg is

  procedure gui_fuel_on_hand_124(
    diesel_qty_available: in integer;
    mogas_qty_available: in integer;
    jet_qty_available: in integer ) is

  begin

    null;  -- the interface would be implemented here

  end gui_fuel_on_hand_124;

end gui_fuel_on_hand_124_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.gui_acc_officer_179.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package simulates the interface for the accountable
--              officer to view the monthly fuel report.
--
--------------------------------------------------------------------------------


package gui_acc_officer_179_pkg is

  procedure gui_acc_officer_179(
    tolerance_jet: in boolean;
    tolerance_mg: in boolean;
    tolerance_df: in boolean );
end gui_acc_officer_179_pkg;

package body gui_acc_officer_179_pkg is

  procedure gui_acc_officer_179(
    tolerance_jet: in boolean;
    tolerance_mg: in boolean;
    tolerance_df: in boolean ) is

  begin

    null;  -- the interface would be implemented here

  end gui_acc_officer_179;

end gui_acc_officer_179_pkg;
```

```
-------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.bulk_rcpt_processor_198.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package serves as a preprocessor.  Insures the bulk
--               receipt input parameters of type of fuel and quantity
--               arrive to be processed at the same time by using a by all
--               triggering constraint.  Arrival of both parameters cause
--               an enable parameter to be generated indicating that the
--               bulk receipt parameters are present and ready to be
--               processed by the receipt processor.
--
-------------------------------------------------------------------------------

package bulk_rcpt_processor_198_pkg is

  procedure bulk_rcpt_processor_198(
    bulk_rcpt_fuel_type: in integer;
    bulk_rcpt_qty: in integer;
    oth_rcpt_enable: in out boolean;
    bulk_rcpt_enable: out boolean );
end bulk_rcpt_processor_198_pkg;

package body bulk_rcpt_processor_198_pkg is

  procedure bulk_rcpt_processor_198(
    bulk_rcpt_fuel_type: in integer;
    bulk_rcpt_qty: in integer;
    oth_rcpt_enable: in out boolean;
    bulk_rcpt_enable: out boolean ) is

  begin

    -- type of fuel and quantity parameters are present for processing

    bulk_rcpt_enable := True;

    -- insure other enable is deactivated

    oth_rcpt_enable := False;

  end bulk_rcpt_processor_198;

end bulk_rcpt_processor_198_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.oth_rcpt_processor_207.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is a preprocessor.  Insures  the other receipt
--              input parameters of fuel type and quantity arrive to be
--              processed at the same time using a by all triggering
--              constraint.  Arrival of both parameters cause an enable
--              parameter to be generated indicating that the other receipt
--              parameters are present and ready to be processed by the
--              receipt processor.
--
--------------------------------------------------------------------------------


package oth_rcpt_processor_207_pkg is

  procedure oth_rcpt_processor_207(
    oth_rcpt_qty: in integer;
    oth_rcpt_fuel_type: in integer;
    bulk_rcpt_enable: in out boolean;
    oth_rcpt_enable: out boolean );
end oth_rcpt_processor_207_pkg;

package body oth_rcpt_processor_207_pkg is

  procedure oth_rcpt_processor_207(
    oth_rcpt_qty: in integer;
    oth_rcpt_fuel_type: in integer;
    bulk_rcpt_enable: in out boolean;
    oth_rcpt_enable: out boolean ) is

  begin

    -- type of fuel and quantity parameters are present for processing

    oth_rcpt_enable := True;

    -- insure other enable is deactivated

    bulk_rcpt_enable := False;

  end oth_rcpt_processor_207;

end oth_rcpt_processor_207_pkg;
```

```
-------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.rcpt_processor_210.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is  the processor of all bulk and other fuel
--              receipts.  Based upon an enable signal and the type of fuel,
--              the processor passes the received quantity of fuel to the
--              appropriate fuel storage tank and totalizer.
--
-------------------------------------------------------------------------------


package rcpt_processor_210_pkg is

  procedure rcpt_processor_210(
    bulk_rcpt_enable: in boolean;
    oth_rcpt_enable: in boolean;
    bulk_rcpt_fuel_type: in integer;
    bulk_rcpt_qty: in integer;
    oth_rcpt_fuel_type: in integer;
    oth_rcpt_qty: in integer;
    diesel_rcpt_qty: out integer;
    mogas_rcpt_qty: out integer;
    jet_rcpt_qty: out integer;
    r_df_qty: out integer;
    r_mg_qty: out integer;
    r_jet_qty: out integer );
end rcpt_processor_210_pkg;

package body rcpt_processor_210_pkg is

  diesel: Constant Integer := 1;
  mogas:  Constant Integer := 2;
  jet:    Constant Integer := 3;

  procedure rcpt_processor_210(
    bulk_rcpt_enable: in boolean;
    oth_rcpt_enable: in boolean;
    bulk_rcpt_fuel_type: in integer;
    bulk_rcpt_qty: in integer;
    oth_rcpt_fuel_type: in integer;
    oth_rcpt_qty: in integer;
    diesel_rcpt_qty: out integer;
    mogas_rcpt_qty: out integer;
    jet_rcpt_qty: out integer;
    r_df_qty: out integer;
    r_mg_qty: out integer;
    r_jet_qty: out integer ) is

  begin

  If bulk_rcpt_enable then

        If bulk_rcpt_fuel_type = diesel then
```

```
            r_df_qty := bulk_rcpt_qty;
            diesel_rcpt_qty := bulk_rcpt_qty;
        End if;

        If bulk_rcpt_fuel_type = mogas then
            r_mg_qty := bulk_rcpt_qty;
            mogas_rcpt_qty := bulk_rcpt_qty;
        End if;

        If bulk_rcpt_fuel_type = jet then
            r_jet_qty := bulk_rcpt_qty;
            jet_rcpt_qty := bulk_rcpt_qty;
        End if;

    End if;

    If oth_rcpt_enable then

        If oth_rcpt_fuel_type = diesel then
            r_df_qty := oth_rcpt_qty;
            diesel_rcpt_qty := oth_rcpt_qty;
        End if;

        If oth_rcpt_fuel_type = mogas then
            r_mg_qty := oth_rcpt_qty;
            mogas_rcpt_qty := oth_rcpt_qty;
        End if;

        If oth_rcpt_fuel_type = jet then
            r_jet_qty := oth_rcpt_qty;
            jet_rcpt_qty := oth_rcpt_qty;
        End if;

    End if;

    end rcpt_processor_210;

end rcpt_processor_210_pkg;
```

```
-----------------------------------------------------------------------------
--
-- File:  fuel_subsystem.df_rcpt_totalizer_274.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is a counter of the daily quantity of diesel
--               fuel received.  This package also provides a hook for
--               future system enhancement such as providing user views
--               of the current daily total receipts of diesel fuel
--               on demand.
--
-----------------------------------------------------------------------------


package df_rcpt_totalizer_274_pkg is

  procedure df_rcpt_totalizer_274(
    df_rcpt_total: in out integer;
    r_df_qty: in integer );
end df_rcpt_totalizer_274_pkg;

package body df_rcpt_totalizer_274_pkg is

  procedure df_rcpt_totalizer_274(
    df_rcpt_total: in out integer;
    r_df_qty: in integer ) is

  begin

   -- adds the new diesel receipt quantity to the daily diesel receipt total

   df_rcpt_total := df_rcpt_total + r_df_qty;

  end df_rcpt_totalizer_274;

end df_rcpt_totalizer_274_pkg;
```

```
------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.mg_rcpt_totalizer_277.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is a counter of the daily quantity of mogas
--               received.  Also provides a hook for future system
--               enhancement such as user views of the current daily total
--               receipts of mogas on demand.
--
------------------------------------------------------------------------------


package mg_rcpt_totalizer_277_pkg is

  procedure mg_rcpt_totalizer_277(
    mg_rcpt_total: in out integer;
    r_mg_qty: in integer );
end mg_rcpt_totalizer_277_pkg;

package body mg_rcpt_totalizer_277_pkg is

  procedure mg_rcpt_totalizer_277(
    mg_rcpt_total: in out integer;
    r_mg_qty: in integer ) is

  begin

    -- adds the new mogas receipt quantity to the daily mogas receipt total

    mg_rcpt_total := mg_rcpt_total + r_mg_qty;

  end mg_rcpt_totalizer_277;

end mg_rcpt_totalizer_277_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.jet_rcpt_totalizer_280.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package  is a counter of the daily quantity of jet
--              fuel received.  Also provides a hook for future system
--              enhancement such as user views of the current daily
--              total receipts of jet fuel on demand.
--
--------------------------------------------------------------------------------


package jet_rcpt_totalizer_280_pkg is

 procedure jet_rcpt_totalizer_280(
  jet_rcpt_total: in out integer;
  r_jet_qty: in integer );
end jet_rcpt_totalizer_280_pkg;

package body jet_rcpt_totalizer_280_pkg is

 procedure jet_rcpt_totalizer_280(
  jet_rcpt_total: in out integer;
  r_jet_qty: in integer ) is

 begin

  -- adds the new jet receipt quantity to the daily jet receipt total

  jet_rcpt_total := jet_rcpt_total + r_jet_qty;

 end jet_rcpt_totalizer_280;

end jet_rcpt_totalizer_280_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.gui_bulk_receipt_3.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package simulates the interface that allows a petroleum
--          specialist to manulally input parameters extracted from DD
--          Form 1348-1 during the receipt of bulk petroleum.  The
--          petroleum specialist enters the type of fuel received,
--          quantity in gallons, and the document number as required
--          by Army Regulation 710-2.
--
--------------------------------------------------------------------------------


with text_string_pkg; use text_string_pkg;

package gui_bulk_receipt_3_pkg is

  procedure gui_bulk_receipt_3(
    bulk_rcpt_fuel_type: out integer;
    bulk_rcpt_doc_number: out text_string;
    bulk_rcpt_qty: out integer );
end gui_bulk_receipt_3_pkg;

package body gui_bulk_receipt_3_pkg is

  procedure gui_bulk_receipt_3(
    bulk_rcpt_fuel_type: out integer;
    bulk_rcpt_doc_number: out text_string;
    bulk_rcpt_qty: out integer ) is

  begin

    null;  -- the interface would be implemented here

  end gui_bulk_receipt_3;

end gui_bulk_receipt_3_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.oth_iss_processor_307.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is a preprocessor.  Insures the equipment issue
--          input parameters of fuel type and quantity arrive to be
--          processed at the same time by using a by all triggering
--          constraint.  Arrival of both parameters cause an enable
--          parameter to be generated indicating that the equipment
--          issue parameters are present and ready to be processed by
--          the issue processor.
--
--------------------------------------------------------------------------------


package oth_iss_processor_307_pkg is

  procedure oth_iss_processor_307(
    eq_iss_qty: in integer;
    eq_iss_fuel_type: in integer;
    bulk_iss_enable: in out boolean;
    oth_iss_enable: out boolean );
end oth_iss_processor_307_pkg;

package body oth_iss_processor_307_pkg is

  procedure oth_iss_processor_307(
    eq_iss_qty: in integer;
    eq_iss_fuel_type: in integer;
    bulk_iss_enable: in out boolean;
    oth_iss_enable: out boolean ) is

  begin

    -- type of fuel and quantity parameters are present for processing

    oth_iss_enable := True;

    -- insure other enable is deactivated

    bulk_iss_enable := False;

  end oth_iss_processor_307;

end oth_iss_processor_307_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.bulk_iss_processor_310.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package serves as a preprocessor.  Insures the
--              bulk issue parameters of fuel type and quantity arrive to
--              be processed at the same time by using a by all triggering
--              constraint.  Arrival of both parameters cause an enable
--              parameter to be generated indicating that the bulk issue
--              parameters are present and ready to be processed by
--              the issue processor.
--
--------------------------------------------------------------------------------


package bulk_iss_processor_310_pkg is

  procedure bulk_iss_processor_310(
    bulk_iss_qty: in integer;
    bulk_iss_fuel_type: in integer;
    oth_iss_enable: in out boolean;
    bulk_iss_enable: out boolean );
end bulk_iss_processor_310_pkg;

package body bulk_iss_processor_310_pkg is

  procedure bulk_iss_processor_310(
    bulk_iss_qty: in integer;
    bulk_iss_fuel_type: in integer;
    oth_iss_enable: in out boolean;
    bulk_iss_enable: out boolean ) is

  begin

    -- type of fuel and quantity parameters are present for processing

    bulk_iss_enable := True;

    -- insure other enable is deactivated

    oth_iss_enable := False;

  end bulk_iss_processor_310;

end bulk_iss_processor_310_pkg;
```

```
-----------------------------------------------------------------------------
--
-- File:  fuel_subsystem.iss_processor_323.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is the processor of all bulk and equipment fuel
--              issues.  Based upon an enable signal and the type of fuel,
--              the processor passes the issued quantity of fuel to the
--              appropriate fuel storage tank and totalizer.
--
-----------------------------------------------------------------------------


package iss_processor_323_pkg is

 procedure iss_processor_323(
   bulk_iss_qty: in integer;
   bulk_iss_fuel_type: in integer;
   eq_iss_qty: in integer;
   eq_iss_fuel_type: in integer;
   oth_iss_enable: in boolean;
   bulk_iss_enable: in boolean;
   jet_iss_qty: out integer;
   mogas_iss_qty: out integer;
   diesel_iss_qty: out integer;
   i_jet_qty: out integer;
   i_mg_qty: out integer;
   i_df_qty: out integer );
end iss_processor_323_pkg;

package body iss_processor_323_pkg is

 diesel: Constant Integer := 1;
 mogas:  Constant Integer := 2;
 jet:    Constant Integer := 3;

 procedure iss_processor_323(
   bulk_iss_qty: in integer;
   bulk_iss_fuel_type: in integer;
   eq_iss_qty: in integer;
   eq_iss_fuel_type: in integer;
   oth_iss_enable: in boolean;
   bulk_iss_enable: in boolean;
   jet_iss_qty: out integer;
   mogas_iss_qty: out integer;
   diesel_iss_qty: out integer;
   i_jet_qty: out integer;
   i_mg_qty: out integer;
   i_df_qty: out integer ) is

 begin

 If bulk_iss_enable then

        If bulk_iss_fuel_type = diesel then
```

```
                i_df_qty := bulk_iss_qty;
                diesel_iss_qty := bulk_iss_qty;
        End if;

        If bulk_iss_fuel_type = mogas then
                i_mg_qty := bulk_iss_qty;
                mogas_iss_qty := bulk_iss_qty;
        End if;

        If bulk_iss_fuel_type = jet then
                i_jet_qty := bulk_iss_qty;
                jet_iss_qty := bulk_iss_qty;
        End if;

    End if;

    If oth_iss_enable then

        If eq_iss_fuel_type = diesel then
                i_df_qty := eq_iss_qty;
                diesel_iss_qty := eq_iss_qty;
        End if;

        If eq_iss_fuel_type = mogas then
                i_mg_qty := eq_iss_qty;
                mogas_iss_qty := eq_iss_qty;
        End if;

        If eq_iss_fuel_type = jet then
                i_jet_qty := eq_iss_qty;
                jet_iss_qty := eq_iss_qty;
        End if;

    End if;

    end iss_processor_323;

end iss_processor_323_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.jet_iss_totalizer_346.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is a counter of the daily quantity of jet fuel
--              issued.  Also provides a hook for future system enhancement
--              such as user views of the current daily total issues of
--              jet fuel on demand.
--
--------------------------------------------------------------------------------


package jet_iss_totalizer_346_pkg is

  procedure jet_iss_totalizer_346(
    jet_iss_total: in out integer;
    i_jet_qty: in integer );
end jet_iss_totalizer_346_pkg;

package body jet_iss_totalizer_346_pkg is

  procedure jet_iss_totalizer_346(
    jet_iss_total: in out integer;
    i_jet_qty: in integer ) is

  begin

    -- adds the new jet issue quantity to the daily jet issue total

    jet_iss_total := jet_iss_total + i_jet_qty;

  end jet_iss_totalizer_346;

end jet_iss_totalizer_346_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.mg_iss_totalizer_349.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This  package is a counter of the daily quantity of mogas
--              issued.  Also provides a hook for future system enhancement
--              such as user views of the current daily total issues of
--              mogas on demnand.
--
--------------------------------------------------------------------------------


package mg_iss_totalizer_349_pkg is

  procedure mg_iss_totalizer_349(
    mg_iss_total: in out integer;
    i_mg_qty: in integer );
end mg_iss_totalizer_349_pkg;

package body mg_iss_totalizer_349_pkg is

  procedure mg_iss_totalizer_349(
    mg_iss_total: in out integer;
    i_mg_qty: in integer ) is

  begin

    -- adds the new mogas issue quantity to the daily mogas issue total

    mg_iss_total := mg_iss_total + i_mg_qty;

  end mg_iss_totalizer_349;

end mg_iss_totalizer_349_pkg;
```

```
-------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.df_iss_totalizer_352.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is a counter of the daily quantity of diesel
--               fuel issued.  This package is also a hook for future system
--               enhancement such as providing user views of the current
--               daily total issues of diesel fuel on demand.
--
-------------------------------------------------------------------------------


package df_iss_totalizer_352_pkg is

  procedure df_iss_totalizer_352(
    df_iss_total: in out integer;
    i_df_qty: in integer );
end df_iss_totalizer_352_pkg;

package body df_iss_totalizer_352_pkg is

  procedure df_iss_totalizer_352(
    df_iss_total: in out integer;
    i_df_qty: in integer ) is

  begin

    -- adds the new diesel issue quantity to the daily diesel issue total

    df_iss_total := df_iss_total + i_df_qty;

  end df_iss_totalizer_352;

end df_iss_totalizer_352_pkg;
```

```
-------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.daily_reporter_410.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package provides a periodic operation that forwards
--              the daily total receipts and issues per fuel type every
--              1 hour in this prototype. The normal period of this
--              operator would be 24 hours in the developed system.
--
-------------------------------------------------------------------------------


package daily_reporter_410_pkg is

  procedure daily_reporter_410(
    df_rcpt_total: in out integer;
    mg_rcpt_total: in out integer;
    jet_rcpt_total: in out integer;
    df_iss_total: in out integer;
    mg_iss_total: in out integer;
    jet_iss_total: in out integer;
    daily_df_rcpt_total: out integer;
    daily_mg_rcpt_total: out integer;
    daily_jet_rcpt_total: out integer;
    daily_df_iss_total: out integer;
    daily_mg_iss_total: out integer;
    daily_jet_iss_total: out integer );
end daily_reporter_410_pkg;

package body daily_reporter_410_pkg is

  procedure daily_reporter_410(
    df_rcpt_total: in out integer;
    mg_rcpt_total: in out integer;
    jet_rcpt_total: in out integer;
    df_iss_total: in out integer;
    mg_iss_total: in out integer;
    jet_iss_total: in out integer;
    daily_df_rcpt_total: out integer;
    daily_mg_rcpt_total: out integer;
    daily_jet_rcpt_total: out integer;
    daily_df_iss_total: out integer;
    daily_mg_iss_total: out integer;
    daily_jet_iss_total: out integer ) is

  begin

    -- forwards daily receipt totals

    daily_df_rcpt_total := df_rcpt_total;
    daily_mg_rcpt_total := mg_rcpt_total;
    daily_jet_rcpt_total := jet_rcpt_total;

    -- forwards daily issue totals
```

```
    daily_df_iss_total := df_iss_total;
    daily_mg_iss_total := mg_iss_total;
    daily_jet_iss_total := jet_iss_total;

    -- reinitialize daily counters

    df_rcpt_total  := 0;
    mg_rcpt_total  := 0;
    jet_rcpt_total := 0;
    df_iss_total   := 0;
    mg_iss_total   := 0;
    jet_iss_total  := 0;

  end daily_reporter_410;

end daily_reporter_410_pkg;
```

```
-------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.bulk_rcpt_db_table_495.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is designed to simulate the storage of the
--               input parameters of a bulk fuel receipt in a relational
--               database table.  The table provides a historical audit trail
--               of all bulk fuel receipts.  Also, this package provides a
--               hook for future system enhancement such as data mining
--               and statistical analysis applications.
--
-------------------------------------------------------------------------------

with text_string_pkg; use text_string_pkg;

package bulk_rcpt_db_table_495_pkg is

  procedure bulk_rcpt_db_table_495(
    bulk_rcpt_fuel_type: in integer;
    bulk_rcpt_qty: in integer;
    bulk_rcpt_doc_number: in text_string );
end bulk_rcpt_db_table_495_pkg;

package body bulk_rcpt_db_table_495_pkg is

  -- defines columns in relational datbase table

  bulk_r_db_fuel_type : Integer;
  bulk_r_db_quantity  : Integer;
  bulk_r_db_doc_num   : text_string;

  procedure bulk_rcpt_db_table_495(
    bulk_rcpt_fuel_type: in integer;
    bulk_rcpt_qty: in integer;
    bulk_rcpt_doc_number: in text_string ) is

  begin

    -- simulates placing parameters in database table columns

    bulk_r_db_fuel_type := bulk_rcpt_fuel_type;
    bulk_r_db_quantity := bulk_rcpt_qty;
    bulk_r_db_doc_num := bulk_rcpt_doc_number;

  end bulk_rcpt_db_table_495;

end bulk_rcpt_db_table_495_pkg;
```

```
-----------------------------------------------------------------------------
--
-- File:  fuel_subsystem.bulk_iss_db_table_498.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is designed to simulate the storage of
--              the input parameters of a bulk fuel issue in a relational
--              database table.  The table provides a historical audit trail
--              of all bulk fuel issues.  The table also provides a hook for
--              future system enhancements such as data mining and
--              statistical analysis applications.
--
-----------------------------------------------------------------------------
with text_string_pkg; use text_string_pkg;

package bulk_iss_db_table_498_pkg is

  procedure bulk_iss_db_table_498(
    bulk_iss_fuel_type: in integer;
    bulk_iss_qty: in integer;
    bulk_iss_doc_num: in text_string;
    bulk_rcv_unit: in text_string;
    bulk_rcv_name: in text_string );
end bulk_iss_db_table_498_pkg;

package body bulk_iss_db_table_498_pkg is

  -- defines columns in relational database table

  bulk_i_db_fuel_type : Integer;
  bulk_i_db_quantity  : Integer;
  bulk_i_db_doc_num   : text_string;
  bulk_i_db_unit      : text_string;
  bulk_i_db_name      : text_string;

  procedure bulk_iss_db_table_498(
    bulk_iss_fuel_type: in integer;
    bulk_iss_qty: in integer;
    bulk_iss_doc_num: in text_string;
    bulk_rcv_unit: in text_string;
    bulk_rcv_name: in text_string ) is

  begin

    -- simulates placing parameters in database table columns

    bulk_i_db_fuel_type := bulk_iss_fuel_type;
    bulk_i_db_quantity := bulk_iss_qty;
    bulk_i_db_doc_num := bulk_iss_doc_num;
    bulk_i_db_unit := bulk_rcv_unit;
    bulk_i_db_name := bulk_rcv_name;

  end bulk_iss_db_table_498;

end bulk_iss_db_table_498_pkg;
```

```
-----------------------------------------------------------------------------
--
-- File:  fuel_subsystem.other_rcpt_db_table_501.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is designed to simulate the storage of the input
--              parameters of all receipts other than bulk in a relational
--              database table.  The table provides a historical audit trail
--              of all receipts other than bulk.  The table also provides a
--              hook for future system enhancements such as data mining
--              and statistical analysis applications.
--
-----------------------------------------------------------------------------


with text_string_pkg; use text_string_pkg;

package other_rcpt_db_table_501_pkg is

  procedure other_rcpt_db_table_501(
    oth_rcpt_fuel_type: in integer;
    oth_rcpt_qty: in integer;
    oth_rcpt_source_id: in text_string;
    oth_rcpt_source_unit: in text_string );
end other_rcpt_db_table_501_pkg;

package body other_rcpt_db_table_501_pkg is

  -- defines columns in relational database table

  oth_r_db_fuel_type   : Integer;
  oth_r_db_quantity    : Integer;
  oth_r_db_source_id   : text_string;
  oth_r_db_source_unit : text_string;

  procedure other_rcpt_db_table_501(
    oth_rcpt_fuel_type: in integer;
    oth_rcpt_qty: in integer;
    oth_rcpt_source_id: in text_string;
    oth_rcpt_source_unit: in text_string ) is

  begin

    -- simulates placing parameters in database table columns

    oth_r_db_fuel_type := oth_rcpt_fuel_type;
    oth_r_db_quantity := oth_rcpt_qty;
    oth_r_db_source_id := oth_rcpt_source_id;
    oth_r_db_source_unit := oth_rcpt_source_unit;

  end other_rcpt_db_table_501;

end other_rcpt_db_table_501_pkg;
```

```
-----------------------------------------------------------------------------
--
-- File:  fuel_subsystem.other_iss_db_table_504.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is designed to simulate the storage of the input
--               parameters for all fuel issues to equipment in a relational
--               database table.  The table provides a historical audit trail
--               of all equipment issues.  The table also provides a hook for
--               future system enhancement such as data mining and
--               statistical analysis applications.
--
-----------------------------------------------------------------------------
with text_string_pkg; use text_string_pkg;

package other_iss_db_table_504_pkg is

  procedure other_iss_db_table_504(
    eq_iss_fuel_type: in integer;
    eq_iss_qty: in integer;
    eq_iss_id: in text_string;
    eq_iss_unit: in text_string;
    eq_iss_name: in text_string );
end other_iss_db_table_504_pkg;

package body other_iss_db_table_504_pkg is

  -- defines columns in relational database table

  oth_i_db_fuel_type : Integer;
  oth_i_db_quantity  : Integer;
  oth_i_db_id        : text_string;
  oth_i_db_unit      : text_string;
  oth_i_db_name      : text_string;

  procedure other_iss_db_table_504(
    eq_iss_fuel_type: in integer;
    eq_iss_qty: in integer;
    eq_iss_id: in text_string;
    eq_iss_unit: in text_string;
    eq_iss_name: in text_string ) is

  begin

    -- simulates placing parameters in database table columns

    oth_i_db_fuel_type := eq_iss_fuel_type;
    oth_i_db_quantity := eq_iss_qty;
    oth_i_db_id := eq_iss_id;
    oth_i_db_unit := eq_iss_unit;
    oth_i_db_name := eq_iss_name;

  end other_iss_db_table_504;

end other_iss_db_table_504_pkg;
```

```
-----------------------------------------------------------------------------
--
-- File:  fuel_subsystem.mo_df_rcpt_totalizer_583.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is a counter of the quantity of diesel fuel
--            received over the course of the month.  Also provides a
--            hook for future system enhancement such as user views of
--            the current total receipts of diesel fuel for the month.
--
-----------------------------------------------------------------------------


package mo_df_rcpt_totalizer_583_pkg is

  procedure mo_df_rcpt_totalizer_583(
    daily_df_rcpt_total: in integer;
    mo_rcpt_df_total: in out integer );
end mo_df_rcpt_totalizer_583_pkg;

package body mo_df_rcpt_totalizer_583_pkg is

  procedure mo_df_rcpt_totalizer_583(
    daily_df_rcpt_total: in integer;
    mo_rcpt_df_total: in out integer ) is

  begin

    -- adds the new daily diesel receipt quantity to the monthly diesel
    -- receipt total

    mo_rcpt_df_total := mo_rcpt_df_total + daily_df_rcpt_total;

  end mo_df_rcpt_totalizer_583;

end mo_df_rcpt_totalizer_583_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.mo_mg_rcpt_totalizer_586.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is a counter of the quantity of mogas received
--              over the course of the month.  Also provides a hook for
--              future system enhancement such as user views of the current
--              total receipts of mogas for the month.
--
--------------------------------------------------------------------------------


package mo_mg_rcpt_totalizer_586_pkg is

  procedure mo_mg_rcpt_totalizer_586(
    daily_mg_rcpt_total: in integer;
    mo_rcpt_mg_total: in out integer );
end mo_mg_rcpt_totalizer_586_pkg;

package body mo_mg_rcpt_totalizer_586_pkg is

  procedure mo_mg_rcpt_totalizer_586(
    daily_mg_rcpt_total: in integer;
    mo_rcpt_mg_total: in out integer ) is

  begin

    -- adds the new daily mogas receipt quantity to the monthly mogas
    -- receipt total

    mo_rcpt_mg_total := mo_rcpt_mg_total + daily_mg_rcpt_total;

  end mo_mg_rcpt_totalizer_586;

end mo_mg_rcpt_totalizer_586_pkg;
```

```
-------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.mo_jet_rcpt_totalizer_589.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is a counter of jet fuel received over the
--              course of the month.  Also provides a hook for future system
--              enhancement such as user views of the current total receipts
--              of jet fuel for the month.
--
-------------------------------------------------------------------------------


package mo_jet_rcpt_totalizer_589_pkg is

  procedure mo_jet_rcpt_totalizer_589(
    daily_jet_rcpt_total: in integer;
    mo_rcpt_jet_total: in out integer );
end mo_jet_rcpt_totalizer_589_pkg;

package body mo_jet_rcpt_totalizer_589_pkg is

  procedure mo_jet_rcpt_totalizer_589(
    daily_jet_rcpt_total: in integer;
    mo_rcpt_jet_total: in out integer ) is

  begin

    -- adds the new daily jet receipt quantity to the monthly jet
    -- receipt total

    mo_rcpt_jet_total := mo_rcpt_jet_total + daily_jet_rcpt_total;

  end mo_jet_rcpt_totalizer_589;

end mo_jet_rcpt_totalizer_589_pkg;
```

```
-------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.mo_df_iss_totalizer_592.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is a counter of the quantity of diesel fuel
--              issued over the course of the month.  Also provides a hook
--              for future system enhancement such as user views of the
--              current total issues of diesel fuel for the month.
--
-------------------------------------------------------------------------------


package mo_df_iss_totalizer_592_pkg is

 procedure mo_df_iss_totalizer_592(
  mo_iss_df_total: in out integer;
  daily_df_iss_total: in integer );
end mo_df_iss_totalizer_592_pkg;

package body mo_df_iss_totalizer_592_pkg is

 procedure mo_df_iss_totalizer_592(
  mo_iss_df_total: in out integer;
  daily_df_iss_total: in integer ) is

 begin

  -- adds the new daily diesel issue quantity to the monthly diesel
  -- issue total

  mo_iss_df_total := mo_iss_df_total + daily_df_iss_total;

 end mo_df_iss_totalizer_592;

end mo_df_iss_totalizer_592_pkg;
```

```
-------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.mo_mg_iss_totalizer_595.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package  is a counter of the quantity of mogas issued
--               over the course of the month.  Also provides a hook for
--               future system enhancement such as user views of the current
--               total issues of mogas for the month.
--
-------------------------------------------------------------------------------


package mo_mg_iss_totalizer_595_pkg is

  procedure mo_mg_iss_totalizer_595(
    mo_iss_mg_total: in out integer;
    daily_mg_iss_total: in integer );
end mo_mg_iss_totalizer_595_pkg;

package body mo_mg_iss_totalizer_595_pkg is

  procedure mo_mg_iss_totalizer_595(
    mo_iss_mg_total: in out integer;
    daily_mg_iss_total: in integer ) is

  begin

    -- adds the new daily mogas issue quantity to the monthly mogas
    -- issue total

    mo_iss_mg_total := mo_iss_mg_total + daily_mg_iss_total;

  end mo_mg_iss_totalizer_595;

end mo_mg_iss_totalizer_595_pkg;
```

```
-------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.mo_jet_iss_totalizer_598.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is a counter of the quantity of jet fuel issued
--              over the course of the month.  Also provides a hook for
--              future system enhancement such as user views of the current
--              total issues of jet fuel for the month.
--
-------------------------------------------------------------------------------


package mo_jet_iss_totalizer_598_pkg is

  procedure mo_jet_iss_totalizer_598(
    mo_iss_jet_total: in out integer;
    daily_jet_iss_total: in integer );
end mo_jet_iss_totalizer_598_pkg;

package body mo_jet_iss_totalizer_598_pkg is

  procedure mo_jet_iss_totalizer_598(
    mo_iss_jet_total: in out integer;
    daily_jet_iss_total: in integer ) is

  begin

    -- adds the new daily jet issue quantity to the monthly jet
    -- issue total

    mo_iss_jet_total := mo_iss_jet_total + daily_jet_iss_total;

  end mo_jet_iss_totalizer_598;

end mo_jet_iss_totalizer_598_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.gui_other_receipt_6.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package simulates the interface that captures all other
--              possible petroleum receipt scenarios i.e. vehicle/aircraft
--              defueling etc.  The interface allows a petroleum specialist
--              to manually enter the type of fuel received, quantity in
--              gallons, an identification number from the source, and
--              the source unit as required by Army Regulation 710-2.
--
--------------------------------------------------------------------------------


with text_string_pkg; use text_string_pkg;

package gui_other_receipt_6_pkg is

  procedure gui_other_receipt_6(
    oth_rcpt_qty: out integer;
    oth_rcpt_source_unit: out text_string;
    oth_rcpt_fuel_type: out integer;
    oth_rcpt_source_id: out text_string );
end gui_other_receipt_6_pkg;

package body gui_other_receipt_6_pkg is

  procedure gui_other_receipt_6(
    oth_rcpt_qty: out integer;
    oth_rcpt_source_unit: out text_string;
    oth_rcpt_fuel_type: out integer;
    oth_rcpt_source_id: out text_string ) is

  begin

    null;  -- the interface would be implemented here

  end gui_other_receipt_6;

end gui_other_receipt_6_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.monthly_reporter_601.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package provides a periodic operation that forwards the
--             monthly total receipts and issues per fuel type every
--             30 hours in this prototype.  The normal period of this
--             operator would be 720 hours, approximately one month, in
--             the developed system.
--
--------------------------------------------------------------------------------


package monthly_reporter_601_pkg is

 procedure monthly_reporter_601(
   mo_iss_jet_total: in out integer;
   mo_iss_mg_total: in out integer;
   mo_iss_df_total: in out integer;
   mo_rcpt_jet_total: in out integer;
   mo_rcpt_mg_total: in out integer;
   mo_rcpt_df_total: in out integer;
   month_df_iss_total: out integer;
   month_df_rcpt_total: out integer;
   month_mg_iss_total: out integer;
   month_mg_rcpt_total: out integer;
   month_jet_rcpt_total: out integer;
   month_jet_iss_total: out integer );
end monthly_reporter_601_pkg;

package body monthly_reporter_601_pkg is

 procedure monthly_reporter_601(
   mo_iss_jet_total: in out integer;
   mo_iss_mg_total: in out integer;
   mo_iss_df_total: in out integer;
   mo_rcpt_jet_total: in out integer;
   mo_rcpt_mg_total: in out integer;
   mo_rcpt_df_total: in out integer;
   month_df_iss_total: out integer;
   month_df_rcpt_total: out integer;
   month_mg_iss_total: out integer;
   month_mg_rcpt_total: out integer;
   month_jet_rcpt_total: out integer;
   month_jet_iss_total: out integer ) is

 begin

   -- forwards monthly receipt totals

   month_df_rcpt_total := mo_rcpt_df_total;
   month_mg_rcpt_total := mo_rcpt_mg_total;
   month_jet_rcpt_total := mo_rcpt_jet_total;
```

```
    -- forwards monthly issue totals

    month_df_iss_total := mo_iss_df_total;
    month_mg_iss_total := mo_iss_mg_total;
    month_jet_iss_total := mo_iss_jet_total;

    -- reinitialize monthly counters

    mo_rcpt_df_total := 0;
    mo_rcpt_mg_total := 0;
    mo_rcpt_jet_total := 0;
    mo_iss_df_total := 0;
    mo_iss_mg_total := 0;
    mo_iss_jet_total := 0;

  end monthly_reporter_601;

end monthly_reporter_601_pkg;
```

```
-----------------------------------------------------------------------------
--
-- File:  fuel_subsystem.daily_rcpt_db_table_740.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is designed to simulate the storage of the
--              daily total receipts per fuel type in a relational
--              database.  The table provides a historical audit trail.
--              The table also provides a hook for future system
--              enhancements such as data mining and statistical analysis.
--
-----------------------------------------------------------------------------


package daily_rcpt_db_table_740_pkg is

  procedure daily_rcpt_db_table_740(
    daily_df_rcpt_total: in integer;
    daily_mg_rcpt_total: in integer;
    daily_jet_rcpt_total: in integer );
end daily_rcpt_db_table_740_pkg;

package body daily_rcpt_db_table_740_pkg is

  -- defines columns in relational database table

  daily_r_db_diesel : Integer;
  daily_r_db_mogas : Integer;
  daily_r_db_jet : Integer;

  procedure daily_rcpt_db_table_740(
    daily_df_rcpt_total: in integer;
    daily_mg_rcpt_total: in integer;
    daily_jet_rcpt_total: in integer ) is

  begin

    -- simulates placing parameters in database table columns

    daily_r_db_diesel := daily_df_rcpt_total;
    daily_r_db_mogas := daily_mg_rcpt_total;
    daily_r_db_jet := daily_jet_rcpt_total;

  end daily_rcpt_db_table_740;

end daily_rcpt_db_table_740_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.daily_iss_db_table_743.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package is designed to simulate the storage of the
--              total daily issues per fuel type in a relational database.
--              The table provides a historical audit trail.  The table
--              also provides a hook for future system enhancements such
--              as data mining and statistical analysis applications.
--
--------------------------------------------------------------------------------


package daily_iss_db_table_743_pkg is

  procedure daily_iss_db_table_743(
    daily_df_iss_total: in integer;
    daily_mg_iss_total: in integer;
    daily_jet_iss_total: in integer );
end daily_iss_db_table_743_pkg;

package body daily_iss_db_table_743_pkg is

  -- defines columns in relational database table

  daily_i_db_diesel : Integer;
  daily_i_db_mogas : Integer;
  daily_i_db_jet : Integer;

  procedure daily_iss_db_table_743(
    daily_df_iss_total: in integer;
    daily_mg_iss_total: in integer;
    daily_jet_iss_total: in integer ) is

  begin

    -- simulates placing parameters in database table columns

    daily_i_db_diesel := daily_df_iss_total;
    daily_i_db_mogas := daily_mg_iss_total;
    daily_i_db_jet := daily_jet_iss_total;

  end daily_iss_db_table_743;

end daily_iss_db_table_743_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.diesel_addition_836.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package simulates the receipt/addition of a
--              quantity of diesel fuel to the storage tank.
--
--------------------------------------------------------------------------------


package diesel_addition_836_pkg is

  procedure diesel_addition_836(
    diesel_volume: in out integer;
    diesel_rcpt_qty: in integer;
    df_qty_on_hand: out integer );
end diesel_addition_836_pkg;

package body diesel_addition_836_pkg is

  procedure diesel_addition_836(
    diesel_volume: in out integer;
    diesel_rcpt_qty: in integer;
    df_qty_on_hand: out integer ) is

  begin

    -- adds the received quantity of diesel fuel to the current volume
    -- of diesel fuel to produce the new quantity of diesel fuel on hand

    df_qty_on_hand := diesel_volume + diesel_rcpt_qty;

    -- updates the current volume of diesel fuel in storage

    diesel_volume := diesel_volume + diesel_rcpt_qty;

  end diesel_addition_836;

end diesel_addition_836_pkg;
```

```
-------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.diesel_subtraction_839.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package simulates the issue/subtraction of a quantity
--              of diesel fuel from the storage tank.
--
-------------------------------------------------------------------------------


package diesel_subtraction_839_pkg is

  procedure diesel_subtraction_839(
    diesel_volume: in out integer;
    diesel_iss_qty: in integer;
    df_qty_on_hand: out integer );
end diesel_subtraction_839_pkg;

package body diesel_subtraction_839_pkg is

  procedure diesel_subtraction_839(
    diesel_volume: in out integer;
    diesel_iss_qty: in integer;
    df_qty_on_hand: out integer ) is

  begin

    -- subtracts the issued quantity of diesel fuel from the current volume
    -- of diesel fuel to produce the new quantity of diesel fuel on hand

    df_qty_on_hand := diesel_volume - diesel_iss_qty;

    -- updates the current volume of diesel fuel in storage

    diesel_volume := diesel_volume - diesel_iss_qty;

  end diesel_subtraction_839;

end diesel_subtraction_839_pkg;
```

```
-------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.diesel_gage_854.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package simulates gaging the diesel storage tank
--              to determine the quantity of fuel on hand. Note: this
--              package can be further enhanced to reflect the effects of
--              volume changes due to environmental factors such as
--              temperature and evaporation.
--
-------------------------------------------------------------------------------


package diesel_gage_854_pkg is

  procedure diesel_gage_854(
    df_qty_on_hand: in integer;
    diesel_qty_available: out integer );
end diesel_gage_854_pkg;

package body diesel_gage_854_pkg is

  procedure diesel_gage_854(
    df_qty_on_hand: in integer;
    diesel_qty_available: out integer ) is

  begin

    -- simulates gaging the diesel tank
    -- if environmental factors were to be included, the df_qty_on_hand
    -- would be increased or decreased accordingly and hence reflected
    -- in the quantity available

    diesel_qty_available := df_qty_on_hand;

  end diesel_gage_854;

end diesel_gage_854_pkg;
```

```
-----------------------------------------------------------------------------
--
-- File:  fuel_subsystem.mogas_addition_888.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package  simulates the receipt/addition of a quantity of
--              mogas to the storage tank.
--
-----------------------------------------------------------------------------


package mogas_addition_888_pkg is

  procedure mogas_addition_888(
    mogas_rcpt_qty: in integer;
    mogas_volume: in out integer;
    mg_qty_on_hand: out integer );
end mogas_addition_888_pkg;

package body mogas_addition_888_pkg is



  procedure mogas_addition_888(
    mogas_rcpt_qty: in integer;
    mogas_volume: in out integer;
    mg_qty_on_hand: out integer ) is

  begin

    -- adds the received quantity of mogas to the current volume of
    -- mogas to produce the new quantity of mogas on hand

    mg_qty_on_hand := mogas_volume + mogas_rcpt_qty;

    -- updates the current volume of mogas in storage

    mogas_volume := mogas_volume + mogas_rcpt_qty;

  end mogas_addition_888;

end mogas_addition_888_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.mogas_subtraction_891.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package simulates the issue/subtraction of a quantity of
--               mogas from the storage tank.
--
--------------------------------------------------------------------------------


package mogas_subtraction_891_pkg is

  procedure mogas_subtraction_891(
    mogas_iss_qty: in integer;
    mogas_volume: in out integer;
    mg_qty_on_hand: out integer );
end mogas_subtraction_891_pkg;

package body mogas_subtraction_891_pkg is

  procedure mogas_subtraction_891(
    mogas_iss_qty: in integer;
    mogas_volume: in out integer;
    mg_qty_on_hand: out integer ) is
  begin

    -- subtracts the issued quantity of mogas from the current volume
    -- of mogas to produce the new quantity of mogas on hand

    mg_qty_on_hand := mogas_volume - mogas_iss_qty;

    -- updates the current volume of mogas in storage

    mogas_volume := mogas_volume - mogas_iss_qty;

  end mogas_subtraction_891;

end mogas_subtraction_891_pkg;
```

```
------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.mogas_gage_894.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package simulates the gaging of the mogas storage tank
--             to determine the quantity of fuel on hand. Note: this package
--             can be further enhanced to reflect the effects of volume
--             changes due to environmental factors such as temperature
--             and evaporation.
--
------------------------------------------------------------------------------


package mogas_gage_894_pkg is

  procedure mogas_gage_894(
    mg_qty_on_hand: in integer;
    mogas_qty_available: out integer );
end mogas_gage_894_pkg;

package body mogas_gage_894_pkg is

  procedure mogas_gage_894(
    mg_qty_on_hand: in integer;
    mogas_qty_available: out integer ) is

  begin

    -- simulates gaging the mogas tank
    -- if environmental factors were included, the mg_qty_on_hand would
    -- be increased or decreased accordingly and hence reflected in the
    -- quantity available

    mogas_qty_available := mg_qty_on_hand;

  end mogas_gage_894;

end mogas_gage_894_pkg;
```

```
------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.gui_bulk_issue_9.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package simulates the interface that allows a petroleum
--              specialist to manually input parameters extracted from
--              DA Form 2765-1 during issue of bulk petroleum.  The petroleum
--              specialist enters the type of fuel issued, quantity in
--              gallons, document number of the issue, the receiving unit,
--              and the name/rank of the receiver as required by
--              Army Regulation 710-2.
--
------------------------------------------------------------------------------

with text_string_pkg; use text_string_pkg;

package gui_bulk_issue_9_pkg is

  procedure gui_bulk_issue_9(
    bulk_rcv_unit: out text_string;
    bulk_iss_doc_num: out text_string;
    bulk_rcv_name: out text_string;
    bulk_iss_qty: out integer;
    bulk_iss_fuel_type: out integer );
end gui_bulk_issue_9_pkg;

package body gui_bulk_issue_9_pkg is

  procedure gui_bulk_issue_9(
    bulk_rcv_unit: out text_string;
    bulk_iss_doc_num: out text_string;
    bulk_rcv_name: out text_string;
    bulk_iss_qty: out integer;
    bulk_iss_fuel_type: out integer ) is

  begin

    null;  -- the interface would be implemented here

  end gui_bulk_issue_9;

end gui_bulk_issue_9_pkg;
```

```
-------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.jet_addition_911.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package simulates the receipt/addition of a quantity of
--              jet fuel to the storage tank.
--
-------------------------------------------------------------------------------


package jet_addition_911_pkg is

  procedure jet_addition_911(
    jet_volume: in out integer;
    jet_rcpt_qty: in integer;
    jet_qty_on_hand: out integer );
end jet_addition_911_pkg;

package body jet_addition_911_pkg is

  procedure jet_addition_911(
    jet_volume: in out integer;
    jet_rcpt_qty: in integer;
    jet_qty_on_hand: out integer ) is

  begin

    -- adds the received quantity of jet fuel to the current volume
    -- of jet fuel to produce the new quantity of jet fuel on hand

    jet_qty_on_hand := jet_volume + jet_rcpt_qty;

    -- updates the current volume of jet fuel in storage

    jet_volume := jet_volume + jet_rcpt_qty;

  end jet_addition_911;

end jet_addition_911_pkg;
```

```
-----------------------------------------------------------------------------
--
-- File:  fuel_subsystem.jet_subtraction_914.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package simulates the issue/subtraction of a quantity of
--             jet fuel from the storage tank.
--
-----------------------------------------------------------------------------


package jet_subtraction_914_pkg is

  procedure jet_subtraction_914(
    jet_volume: in out integer;
    jet_iss_qty: in integer;
    jet_qty_on_hand: out integer );
end jet_subtraction_914_pkg;

package body jet_subtraction_914_pkg is

  procedure jet_subtraction_914(
    jet_volume: in out integer;
    jet_iss_qty: in integer;
    jet_qty_on_hand: out integer ) is

  begin

    -- subtracts the issued quantity of jet fuel from the current volume
    -- of jet fuel to produce the new quantity of jet fuel on hand

    jet_qty_on_hand := jet_volume - jet_iss_qty;

    -- updates the current volume of jet fuel in storage

    jet_volume := jet_volume - jet_iss_qty;

  end jet_subtraction_914;

end jet_subtraction_914_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.jet_gage_917.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package simulates gaging the jet fuel storage tank to
--             determine the quantity of fuel on hand. Note: this package can
--             be further enhanced to reflect the effects of volume changes
--             due to environmental factors such as temperature and
--             evaporation.
--
--------------------------------------------------------------------------------


package jet_gage_917_pkg is

  procedure jet_gage_917(
    jet_qty_on_hand: in integer;
    jet_qty_available: out integer );
end jet_gage_917_pkg;

package body jet_gage_917_pkg is

  procedure jet_gage_917(
    jet_qty_on_hand: in integer;
    jet_qty_available: out integer ) is

  begin

    -- simulates gaging the jet fuel tank
    -- if environmental factors were to be included, the jet_qty_on_hand
    -- would be increased or decreased accordingly and hence reflected in
    -- the quantity available

    jet_qty_available := jet_qty_on_hand;

  end jet_gage_917;

end jet_gage_917_pkg;
```

```
-------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.diesel_iss_acct_proc_934.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package sets the total monthly diesel issues for
--             processing.
--
-------------------------------------------------------------------------------


package diesel_iss_acct_proc_934_pkg is

  procedure diesel_iss_acct_proc_934(
    month_df_iss_total: in integer;
    total_mo_df_iss: out integer );
end diesel_iss_acct_proc_934_pkg;

package body diesel_iss_acct_proc_934_pkg is

  procedure diesel_iss_acct_proc_934(
    month_df_iss_total: in integer;
    total_mo_df_iss: out integer ) is

  begin

    total_mo_df_iss := month_df_iss_total;

  end diesel_iss_acct_proc_934;

end diesel_iss_acct_proc_934_pkg;
```

```
-------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.diesel_rcpt_acct_proc_937.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package sets the total monthly diesel receipts for
--              processing.
--
-------------------------------------------------------------------------------


package diesel_rcpt_acct_proc_937_pkg is

  procedure diesel_rcpt_acct_proc_937(
    month_df_rcpt_total: in integer;
    total_mo_df_rcpt: out integer );
end diesel_rcpt_acct_proc_937_pkg;

package body diesel_rcpt_acct_proc_937_pkg is

  procedure diesel_rcpt_acct_proc_937(
    month_df_rcpt_total: in integer;
    total_mo_df_rcpt: out integer ) is

  begin

    total_mo_df_rcpt := month_df_rcpt_total;

  end diesel_rcpt_acct_proc_937;

end diesel_rcpt_acct_proc_937_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.mogas_iss_acct_proc_940.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package sets the total monthly mogas issues for
--              processing.
--
--------------------------------------------------------------------------------


package mogas_iss_acct_proc_940_pkg is

  procedure mogas_iss_acct_proc_940(
    month_mg_iss_total: in integer;
    total_mo_mg_iss: out integer );
end mogas_iss_acct_proc_940_pkg;

package body mogas_iss_acct_proc_940_pkg is

  procedure mogas_iss_acct_proc_940(
    month_mg_iss_total: in integer;
    total_mo_mg_iss: out integer ) is

  begin

    total_mo_mg_iss := month_mg_iss_total;

  end mogas_iss_acct_proc_940;

end mogas_iss_acct_proc_940_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File: fuel_subsystem.mogas_rcpt_acct_proc_943.a
-- Author: Lawrence A. Kominiak, Major, USA
-- Project: Fuel Automated Subsystem of ICS3
-- Date: February 1998
-- Description: This package sets the total monthly mogas receipts for
--              processing.
--
--------------------------------------------------------------------------------


package mogas_rcpt_acct_proc_943_pkg is

  procedure mogas_rcpt_acct_proc_943(
    month_mg_rcpt_total: in integer;
    total_mo_mg_rcpt: out integer );
end mogas_rcpt_acct_proc_943_pkg;

package body mogas_rcpt_acct_proc_943_pkg is

  procedure mogas_rcpt_acct_proc_943(
    month_mg_rcpt_total: in integer;
    total_mo_mg_rcpt: out integer ) is

  begin

    total_mo_mg_rcpt := month_mg_rcpt_total;

  end mogas_rcpt_acct_proc_943;

end mogas_rcpt_acct_proc_943_pkg;
```

```
-------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.jet_iss_acct_proc_946.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package sets the total monthly jet issues for
--               processing.
--
-------------------------------------------------------------------------------


package jet_iss_acct_proc_946_pkg is

  procedure jet_iss_acct_proc_946(
    month_jet_iss_total: in integer;
    total_mo_jet_iss: out integer );
end jet_iss_acct_proc_946_pkg;

package body jet_iss_acct_proc_946_pkg is

  procedure jet_iss_acct_proc_946(
    month_jet_iss_total: in integer;
    total_mo_jet_iss: out integer ) is

  begin

    total_mo_jet_iss := month_jet_iss_total;

  end jet_iss_acct_proc_946;

end jet_iss_acct_proc_946_pkg;
```

```
-------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.jet_rcpt_acct_proc_949.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package sets the total monthly jet receipts for
--               processing.
--
-------------------------------------------------------------------------------


package jet_rcpt_acct_proc_949_pkg is

  procedure jet_rcpt_acct_proc_949(
    month_jet_rcpt_total: in integer;
    total_mo_jet_rcpt: out integer );
end jet_rcpt_acct_proc_949_pkg;

package body jet_rcpt_acct_proc_949_pkg is

  procedure jet_rcpt_acct_proc_949(
    month_jet_rcpt_total: in integer;
    total_mo_jet_rcpt: out integer ) is

  begin

    total_mo_jet_rcpt := month_jet_rcpt_total;

  end jet_rcpt_acct_proc_949;

end jet_rcpt_acct_proc_949_pkg;
```

```
-------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.df_acct_calc_952.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package determines whether diesel fuel
--              accountability is within tolerance.
--
-------------------------------------------------------------------------------


package df_acct_calc_952_pkg is

 procedure df_acct_calc_952(
   total_mo_df_iss: in integer;
   total_mo_df_rcpt: in integer;
   opening_inv_diesel: in out integer;
   diesel_qty_available: in integer;
   tolerance_df: out boolean );
end df_acct_calc_952_pkg;

package body df_acct_calc_952_pkg is

 df_closing_book_bal : Integer;
 df_mo_gain_loss : Integer;
 df_allow_gain_loss : Integer;

 procedure df_acct_calc_952(
   total_mo_df_iss: in integer;
   total_mo_df_rcpt: in integer;
   opening_inv_diesel: in out integer;
   diesel_qty_available: in integer;
   tolerance_df: out boolean ) is

 begin

   -- determine if monthly diesel accounting is within tolerance

   df_closing_book_bal := opening_inv_diesel + total_mo_df_rcpt - total_mo_df_iss;
   df_mo_gain_loss := df_closing_book_bal - diesel_qty_available;
   df_allow_gain_loss := (opening_inv_diesel + total_mo_df_rcpt) * 5 / 1000;

   If abs(df_mo_gain_loss) > abs(df_allow_gain_loss) then
         tolerance_df := False;
   else
         tolerance_df := True;
   End if;

   -- adjust opening monthly inventory to the physical quantity on hand

   opening_inv_diesel := diesel_qty_available;

 end df_acct_calc_952;

end df_acct_calc_952_pkg;
```

```
------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.mg_acct_calc_955.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package  determines whether mogas fuel accountability is
--               within tolerance.
--
------------------------------------------------------------------------------


package mg_acct_calc_955_pkg is

  procedure mg_acct_calc_955(
    total_mo_mg_iss: in integer;
    total_mo_mg_rcpt: in integer;
    opening_inv_mogas: in out integer;
    mogas_qty_available: in integer;
    tolerance_mg: out boolean );
end mg_acct_calc_955_pkg;

package body mg_acct_calc_955_pkg is

  mg_closing_book_bal : Integer;
  mg_mo_gain_loss : Integer;
  mg_allow_gain_loss : Integer;

  procedure mg_acct_calc_955(
    total_mo_mg_iss: in integer;
    total_mo_mg_rcpt: in integer;
    opening_inv_mogas: in out integer;
    mogas_qty_available: in integer;
    tolerance_mg: out boolean ) is

  begin

    -- determine if monthly mogas accounting is within tolerance

    mg_closing_book_bal := opening_inv_mogas + total_mo_mg_rcpt - total_mo_mg_iss;
    mg_mo_gain_loss := mg_closing_book_bal - mogas_qty_available;
    mg_allow_gain_loss := (opening_inv_mogas + total_mo_mg_rcpt) / 100;

    If abs(mg_mo_gain_loss) > abs(mg_allow_gain_loss) then
         tolerance_mg := False;
    else
         tolerance_mg := True;
    End if;

    -- adjust opening monthly inventory to the physical quantity on hand

    opening_inv_mogas := mogas_qty_available;

  end mg_acct_calc_955;

end mg_acct_calc_955_pkg;
```

```
------------------------------------------------------------------------------
--
-- File:  fuel_subsystem.jet_acct_calc_958.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package determines whether jet fuel accountability is
--              within tolerance.
--
------------------------------------------------------------------------------


package jet_acct_calc_958_pkg is

  procedure jet_acct_calc_958(
    total_mo_jet_iss: in integer;
    total_mo_jet_rcpt: in integer;
    opening_inv_jet: in out integer;
    jet_qty_available: in integer;
    tolerance_jet: out boolean );
end jet_acct_calc_958_pkg;

package body jet_acct_calc_958_pkg is

  jet_closing_book_bal : Integer;
  jet_mo_gain_loss : Integer;
  jet_allow_gain_loss : Integer;

  procedure jet_acct_calc_958(
    total_mo_jet_iss: in integer;
    total_mo_jet_rcpt: in integer;
    opening_inv_jet: in out integer;
    jet_qty_available: in integer;
    tolerance_jet: out boolean ) is

  begin

   -- determine if monthly jet fuel accountability is within tolerance

   jet_closing_book_bal := opening_inv_jet + total_mo_jet_rcpt - total_mo_jet_iss;
   jet_mo_gain_loss := jet_closing_book_bal - jet_qty_available;
   jet_allow_gain_loss := (opening_inv_jet + total_mo_jet_rcpt) / 100;

   If abs(jet_mo_gain_loss) > abs(jet_allow_gain_loss) then
           tolerance_jet := False;
   else
           tolerance_jet := True;
   End If;

   -- adjust opening monthly inventory to the physical quantity on hand

   opening_inv_jet := jet_qty_available;

  end jet_acct_calc_958;

end jet_acct_calc_958_pkg;
```

```
--------------------------------------------------------------------------------
--
-- File:  text_string_pkg.a
-- Author:  Lawrence A. Kominiak, Major, USA
-- Project:  Fuel Automated Subsystem of ICS3
-- Date:  February 1998
-- Description:  This package defines the text_string type, as a
--              string of 100 characters.
--
--------------------------------------------------------------------------------

package text_string_pkg is

  subtype text_string is string(1..100);

end text_string_pkg;
```

APPENDIX G

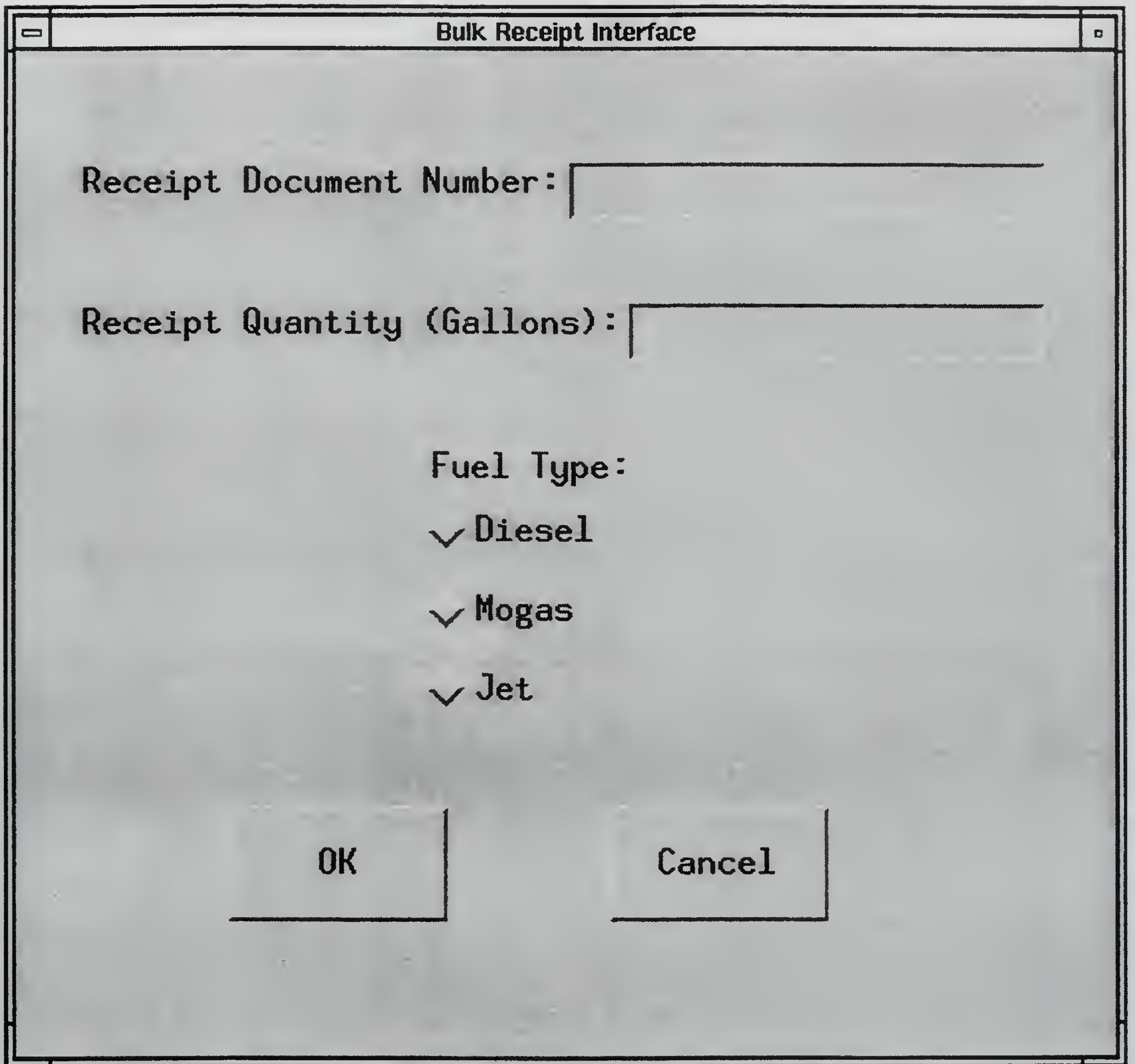


Figure 1. Bulk Receipt Graphical User Interface

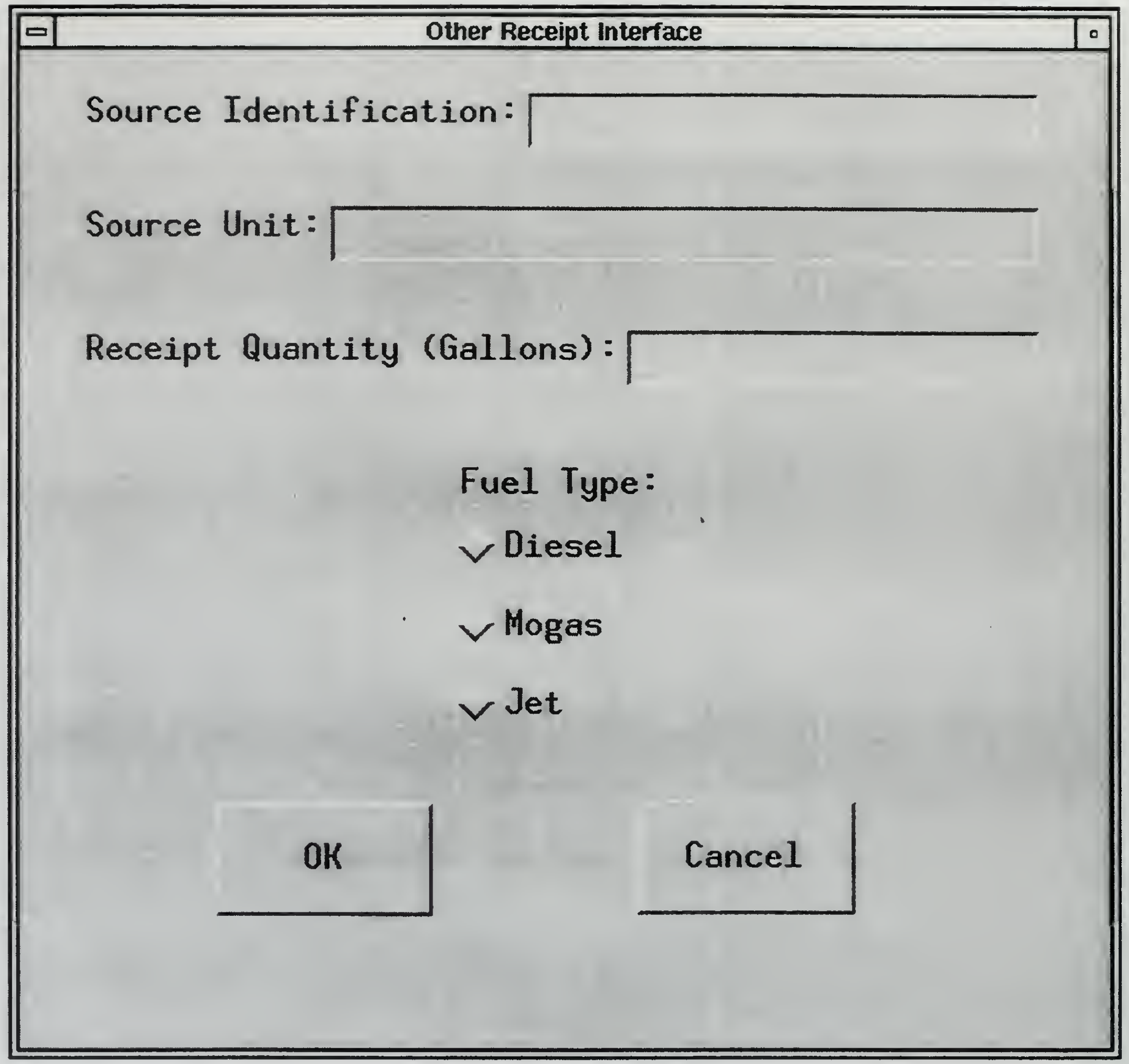


Figure 2. Other Receipt Graphical User Interface

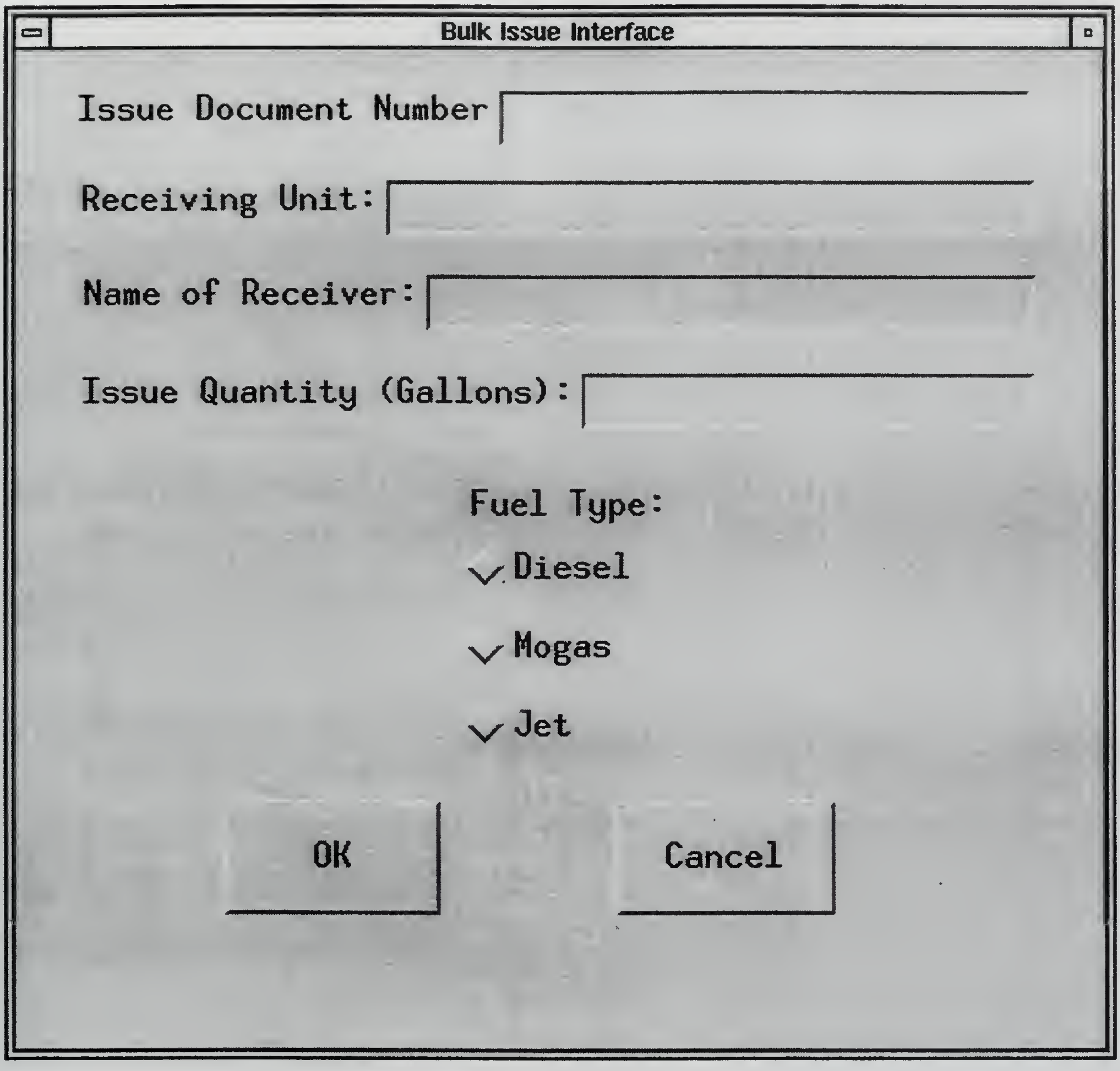


Figure 3. Bulk Issue Graphical User Interface

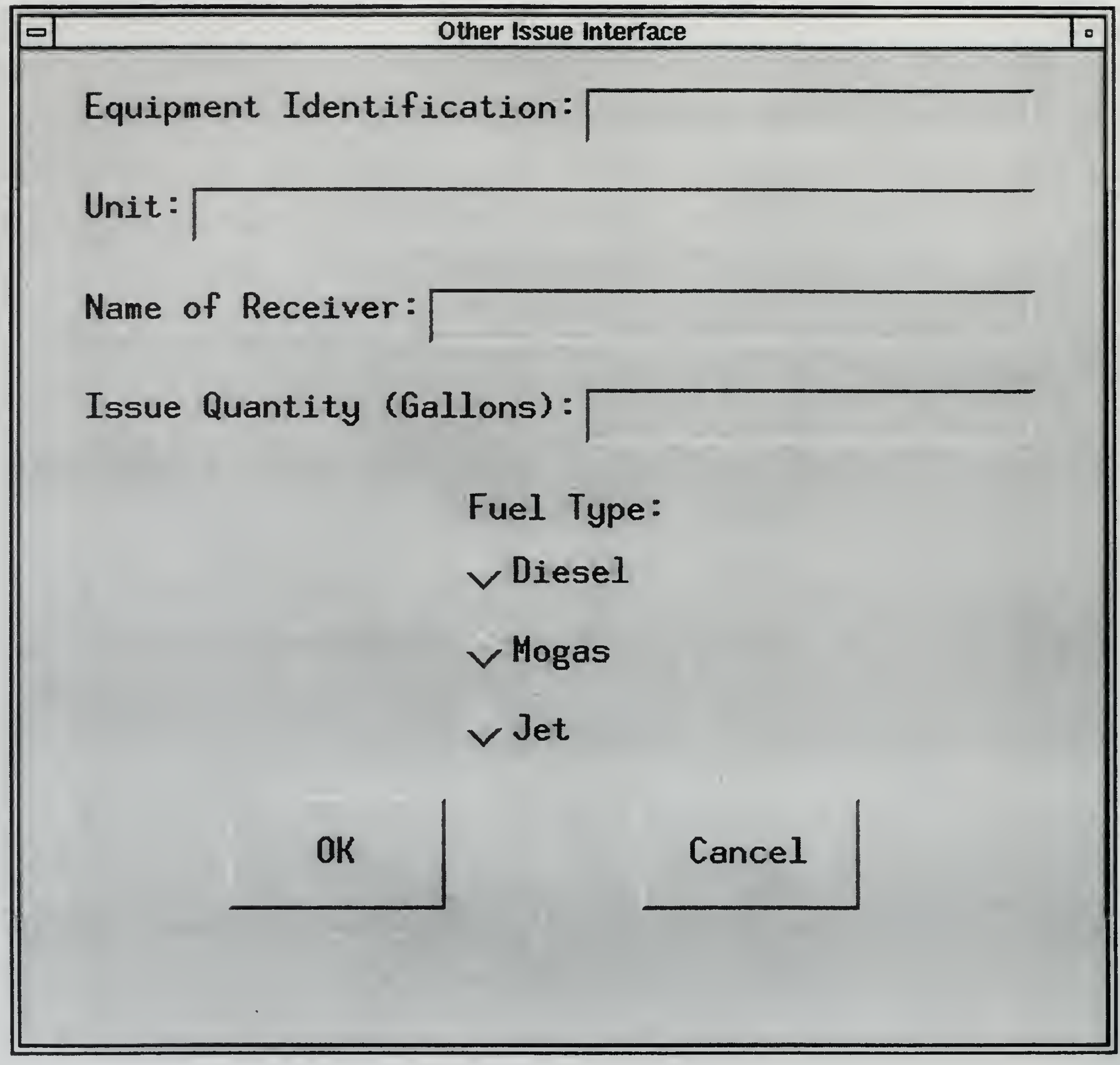


Figure 4. Other Issue Graphical User Interface

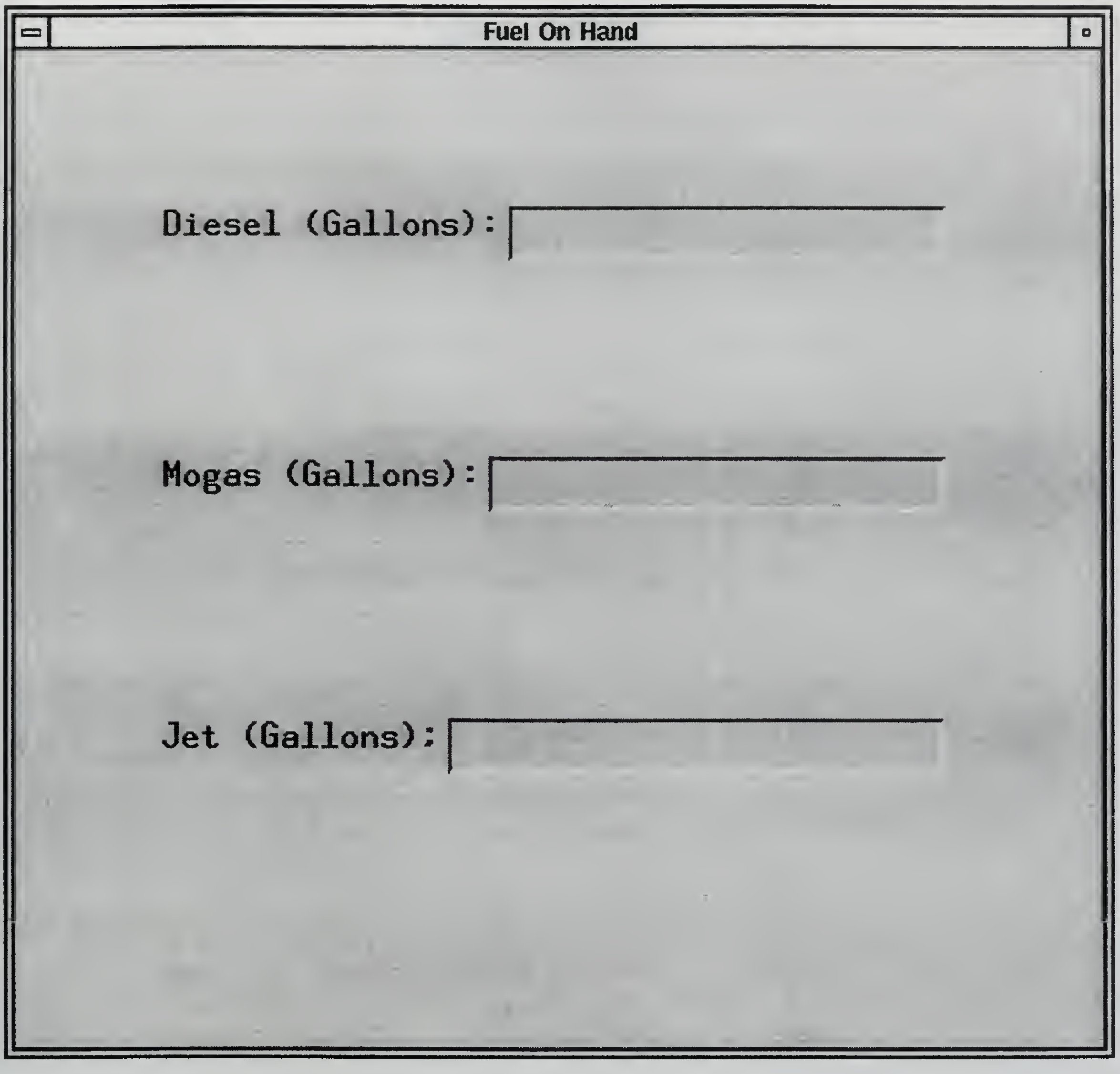


Figure 5. Fuel On Hand Interface

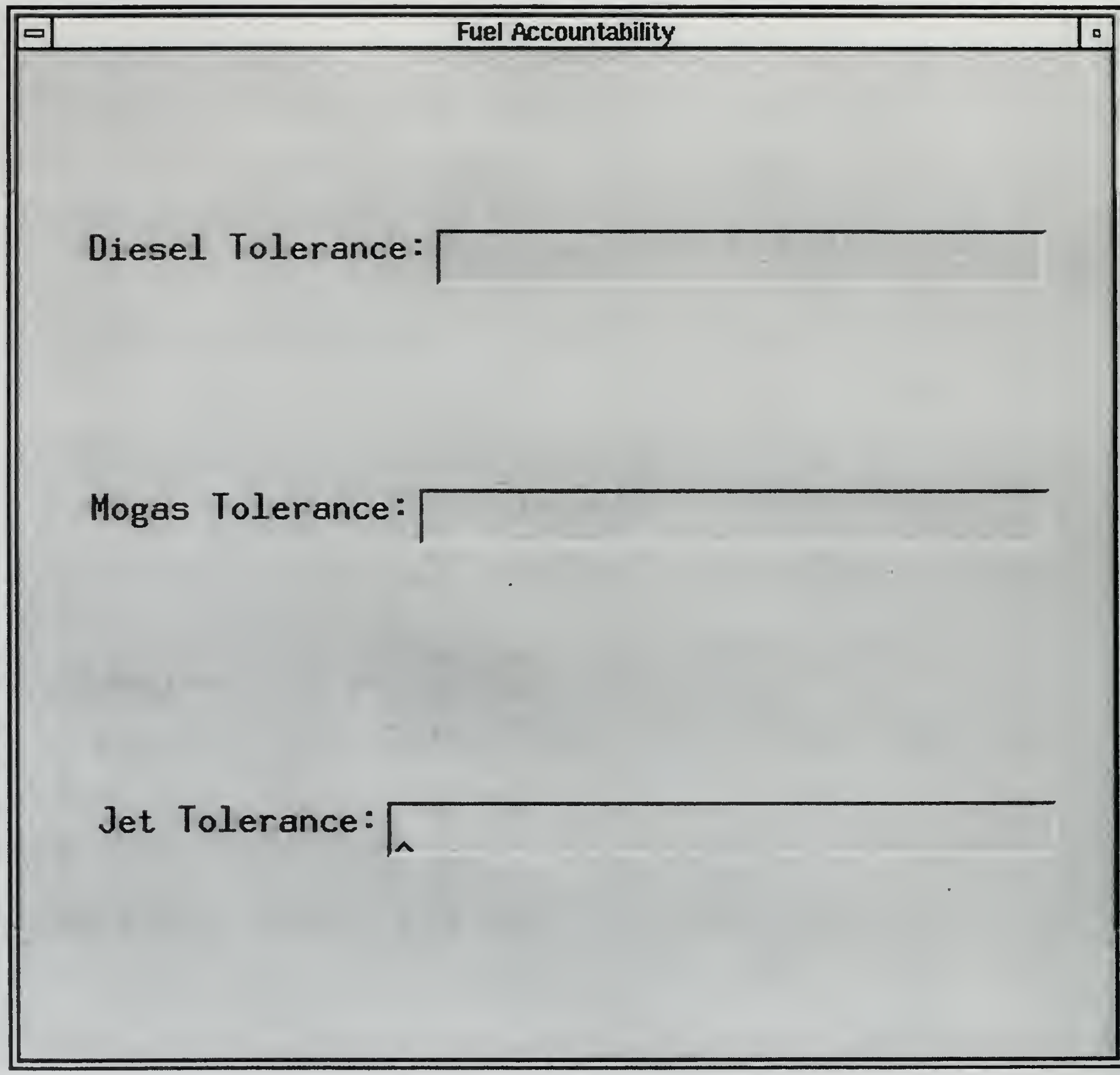


Figure 6. Fuel Accountability Interface

LIST OF REFERENCES

1. Seffers, George I., "Army may devote $28 billion to digitization", *Army Times*, 10 November 1997.

2. Robinson, Thomas W., "Force XXI Combat Service Support", United States Army Combined Arms Support Command and Fort Lee, 1997.

3. U.S. Army, Training and Doctrine Command, *Land Combat in the 21st Century*, Fort Monroe, Virginia, 1997.

4. Joint Chief of Staff Publication 4.0, *Doctrine for Logistic Support of Joint Operations*, Washington, D.C., 25 September 1992.

5. Wilson, Johnnie E., "Leveraging Logistics Technology Towards Force XXI", *Army Logistician*.

6 U.S. Army, Combined Arms Support Command, Integrated Combat Service Support System – ICS3, 1997, http://www.cascom.army.mil/automation/ICS3_Integrated_ Combar_Service_Support_System/Briefings/ICS3_Standard_Briefing.ppt.

7. U.S. Army, Combined Arms Support Command, Integrated Combat Service Support System (ICS3), 1997, http://www.cascom.army.mil/automation/ICS3_Integrated_ Combat_Service_Support_System/Miscellaneuos/Short_descriptionon_of_ICS3.

8. Report HQ 92-A3, *Management of Bulk Petroleum*, U.S. Army Audit Agency, Washington, D.C., 1992.

9. U.S. Navy, Naval Postgraduate School, "CAPS – A Tool for Complex System Development and Acquisition", 1997.

10. Luqi, "Computer Aided Prototyping System (CAPS) Executive Summary Brief", United States Naval Postgraduate School, 1997.

11. Luqi, "Class Notes for CS4520", United States Naval Postgraduate School, 1997.

12. Luqi, "Software Evolution Through Rapid Prototyping", *IEEE Computer*, May 1989.

13. Luqi, "Formal Methods: Promises and Problems", *IEEE Software*, January /February 1997.

14. Luqi, Berzins and Yeh, "A Prototyping Language for Real-Time Software", *IEEE Transactions on Software Engineering*, Vol. 14, No.10, October 1988.

15. Berzins and Luqi, *Software Engineering with Abstractions*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1991.

16. U.S. Navy, Naval Postgraduate School, "CAPS Tutorial", 1996.

BIBLIOGRAPHY

Army Regulation 710-2, *Supply Policy Below Wholesale Level*, Department of the Army, Washington, D.C., 28 February 1994.

Army Regulation 735-5, *Policies and Procedures for Property Accountability*, Department of the Army, Washington, D.C., 28 February 1994.

Barnes, John, *Programming in Ada 95*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1996.

Department of the Army Pamphlet 710-2-1, *Using Unit Supply System Manual Procedures*, Department of the Army, Washington, D.C., 28 February 1994.

Department of the Army Pamphlet 710-2-2, *Supply Support Activity System Manual Procedures*, Department of the Army, Washington, D.C., 28 February 1994.

Defense Fuel Supply Center Fuels Automated System, Defense Fuel Supply Center, 1997, http://www.dfsc.dla.mil/main/s/white.html

Feldman, M.B., Koffman, E.B., *Ada 95 Problem Solving and Program Design*, Addison-Wesley Publishing Company, Reading, Massachusetts,1996.

Field Manual 10-1, *Quartermaster Principles*, Department of the Army, Washington, D.C., August 1994.

Field Manual 10-27, *General Supply in Theaters of Operations*, Department of the Army, Washington, D.C., 20 April 1993.

Field Manual 10-67, *Petroleum Supply in Theaters of Operations*, Department of the Army, Washington, D.C., October 1985.

Field Manual 10-69, *Petroleum Supply Point Equipment and Operation*, Department of the Army, Washington, D.C., October 1989.

Field Manual 54-10, *Logistics: An Overview of the Total System*, Department of the Army, Washington, D.C., April 1993.

Field Manual 101-5-1, *Operational Terms and Symbols*, Department of the Army, Washington, D.C., October 1985.

Field Manual 101-10-1, *Organizational, Technical, and Logistical Data Planning Factors*, Department of the Army, Washington, D.C., October 1987.

Naylor, Sean D., "Digital revolution shows promise but needs proof", *Army Times*, 13 January 1997.

Pexton, Patrick, "Fewer soldiers needed in Army's chip-driven future", *Army Times*, 6 January 1997.

Training and Doctrine Command Pamphlet 525-5, *Force XXI Operations*, Department of the Army, Washington, D.C., February 1997.

Training and Doctrine Command Pamphlet 525-200-6, *Combat Service Support*, Department of the Army, Washington, D.C., 1 August 1994.

U.S. Army, Combined Arms Support Command, Integrated Combat Service Support System ICS3 Phase I Implementation Plan, 1997, http://www.cascom.army.mil/automation/ICS3_Integrated_Combar_Service_Support_System/Briefings/ICS3_Implementation_Plan.ppt.

U.S. Army, Combined Arms Support Command, Information Paper – Integrated Combat Service Support System (ICS3), 1997, http://www.cascom.army.mil/automation/ICS3_Integrated_Combar_Service_Support_System/Master_Plan_Info_Paper_ICS3.doc.

U.S. Army, Combined Arms Support Command, Integrated Combat Service Support System (ICS3) Requirements Definition Questionnaire, 1997, http://www.cascom.army.mil/automation/ICS3_Integrated_Combar_Service_Support_System/Miscellaneous/ICS3_Question.DOC.

U.S. Army, Combined Arms Support Command, CSS Automation in the Information Age, 1997, http://www.cascom.army.mil/automation/ICS3_Integrated_ Combat_Service_Support_System/Briefings/Senior_Leadership_Training_Conference_July_ 1997.ppt.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center...2
   8725 John J. Kingman Road, Ste 0944
   Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, California 93943-5101

3. Defense Logistic Studies Information Exchange...1
   U.S. Army Logistics Management College
   Fort Lee, Virginia 23801-6043

4. Dr. Dan Boger, Chairman, Code CS/Bo...1
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California 93943-5100

5. Dr. Luqi, Code CS/Lq...1
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California 93943-5100

6. Dr. Valdis Berzins Code CS/Vb..1
   Department of Computer Science
   Naval Postgraduate School
   Monterey, CA 93943-5100

7. Major Lawrence Kominiak...2
   930 Greenwood Road
   Teaneck, New Jersey 07666

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

9 483NPG TH 2912
10/99 22527-100

DUDLEY KNOX LIBRARY
3 2768 00364648 0